Title

Development of lean, efficient, and fast physics-framed deep-learning-based proxy models for subsurface carbon storage

Authors

Omosebi, Omotayo A

Oldenburg, Curtis M

Reagan, Matthew

Publication Date

2022-02-01

DOI

10.1016/j.ijggc.2021.103562

Peer reviewed

Development of Lean, Efficient, and Fast Physics-Framed Deep-Learning-Based Proxy Models for Subsurface Carbon Storage

Omotayo A. Omosebi*[,1], Curtis M. Oldenburg[1], Matthew Reagan[1]

[1]Energy Geosciences Division, Lawrence Berkeley National Laboratory

***Corresponding author**

Email: oaomosebi@lbl.gov

Address:

Energy Geosciences Division

Lawrence Berkeley National Laboratory

1 Cyclotron Road,

Berkeley, CA 94720, M/S 74R316C

United states of America

Development of Lean, Efficient, and Fast Physics-Framed Deep-Learning-Based Proxy Models for Subsurface Carbon Storage

Omotayo A. Omosebi*[,1], Curtis M. Oldenburg[1], Matthew Reagan[1]

[1]Energy Geosciences Division, Lawrence Berkeley National Laboratory

ABSTRACT

We present deep-learning-based surrogate models for CCUS developed with four different algorithms and a physics-framed two-phase flow problem involving displacement of water by CO_2. The deep-learning models were trained using 3D data sets describing the pressure plume, CO_2 saturation plume, and water extraction rate generated by numerical simulation. The hyperparameters defining the architecture of the neural networks were optimized to determine the slimmest network size and training parameters that give the most efficient performance at the least training cost. To develop a robust model that closely mimics the governing physical laws, the discretized form of the two-phase fluid transport equation was used to formulate the supervised deep-learning task.

The algorithms investigated in this study predicted the data to above 95% accuracy, with the multi-layer perceptron model demonstrating the best performance by balancing training speed, prediction time, and prediction accuracy with lean network capacity. Furthermore, the surrogate models simultaneously predict reservoir pressure and CO_2 saturation in every grid block, including the surface well extraction rate and bottomhole pressure, at all simulation times for a given static model realization in just a few seconds on a standard desktop computer. A key outcome of this study is that limits can be placed on network design parameters to avoid over

designing neural networks, with associated efficiencies in training and prediction times. This is very useful because large volumes of data may be generated in CCUS projects and over-design of neural network architectures imposes penalties that are antithetical to the goal of near-real time forecasting.

Keywords: Fast Proxy Model, Deep-Learning, Machine-Learning, Physics-guided, Carbon storage, Carbon sequestration

1. INTRODUCTION

Carbon capture, utilization, and storage (CCUS) is the direct injection of anthropogenic carbon dioxide (CO_2) primarily into oil reservoirs for enhanced oil recovery or into deep saline aquifers for permanent storage away from the atmosphere[1, 2]. A number of CCUS projects are active with millions of tons of CO_2 already injected[3]. Despite some success in bringing CCUS projects online, there are many uncertainties and barriers to widespread industrial-scale implementation of CCUS around the world including economic, regulatory, legal, political, and environmental issues. To improve regulatory and social acceptance of CCUS, along with reducing uncertainties that affect the costs of implementing CCUS as a viable technology for mitigating climate change, it is useful to explore ways to simulate subsurface processes and better communicate technical information to quickly resolve stakeholder concerns about a range of issues. These issues include induced seismicity and potential for felt earthquakes, the fate of CO_2 in the subsurface, the quantity of CO_2 that can be practically sequestered, the number of wells needed for large-scale injection, the potential for vertical fluid migration and the risk of CO_2 leakage into freshwater aquifers, and the return on investment for operators. A holistic approach to address these concerns requires exploring new reservoir modeling and simulation techniques that offer much more rapid, near real-time insights into critical subsurface processes that traditionally have required long run times even using the best available computational resources.

Detailed understanding of the expected reservoir pore pressure, the extent of the CO_2 saturation plume, and the changes in geomechanical stress arising from perturbations of the equilibrium state of the reservoir during CO_2 injection are essential for project permitting prior to injection. Following the startup of injection, the same information is critical for operators, regulators, and the public to understand the benefits, risks, and effectiveness of operations throughout the CCUS project life cycle[4, 5]. The state of the reservoir is traditionally modeled through physics-based numerical simulation in which mathematical models describing coupled subsurface processes are solved for reservoir pressure, CO_2 saturation, etc. in both space and

time. The mathematical models typically comprise partial differential equations, along with appropriate boundary and initial conditions, describing conservation of mass and momentum of all phases and their components in the porous or fractured media[6]. These models have been implemented in a number of numerical simulators, e.g., the *TOUGH* family of codes, *CMG-GEM*, Eclipse, and others that have proven very effective at high-fidelity reservoir simulation[7, 8]. During the simulation process, the overall objective is to capture the spatio-temporal evolution of pressure and saturation across the entire simulation domain over which a mesh of grid blocks is constructed. (Note that the term "grid block" arises primarily in integral finite difference- and finite volume-based numerical methods whereas other numerical methods may solve for primary variables at grid points rather than within a "grid block". Nevertheless, for simplicity in this paper and without loss of generality, we will refer in this paper to the points of the numerical mesh at which the primary variables are calculated as being grid blocks.) Depending on the scale of the problem, the numerical solution of the pressure and CO_2 saturation needs to be computed over thousands to millions of grid blocks in space, resulting in computationally expensive simulation of fluid displacement in the reservoir at different snapshots in time. This computational burden is exacerbated by the fact that simulation runs are typically performed on a single geological realization of porosity and permeability fields, the so-called static model. To incorporate inherent uncertainties in the static (hydrostratigraphic) models, multiple models must be generated (e.g., P10, P25, P50, etc.) and multiple simulations must be performed. This standard workflow involving simulating reservoir evolution with the conventional reservoir modeling approaches inhibits fast turnaround of results and does not allow near-real time forecasting.

Proxy models, approximations of computationally expensive full-physics numerical models, are one approach to overcoming the slow execution times of large physics-based models [9-13]. These models typically involve fewer computations, which dramatically reduces computational time during forward modeling. Unlike full-physics models, they are well-suited for rapid sensitivity analysis and optimization during inverse modeling. Deep-learning (DL)-based

proxy modeling is a very attractive option capable of accelerating real-time decisions in subsurface storage of anthropogenic CO_2 by leveraging algorithms that have been developed for machine learning (ML) to learn dynamic patterns in a given subsurface system. For CCUS applications, the objective in surrogate modeling is to reconstruct the physics-based spatio-temporal surfaces of pressure and saturation using neural networks as functional approximations. Using this approach, it is possible to develop a single proxy model capable of simulating multiple representations of the reservoir concurrently at all simulation times. Specifically, reservoir state variables and well injectivity could be rapidly predicted in a single computational step with multiple realizations of porosity and permeability as inputs. This modeling approach uses significantly less time and computational power than running full-physics reservoir simulations, and thus is well-suited for CCUS field development where multiple realizations of the reservoir must be generated to capture geological uncertainties. By taking advantage of the strength of standard deep-learning algorithms, dynamic patterns in the subsurface data could be learned, thereby capturing reservoir processes over time. The learned field patterns are parameterized in weights and biases of neural networks to be used for fast forward modeling.

In this paper, we present deep-learning-based proxy models trained with the multi-layer perceptron, convolutional neural network, long short-term memory, and gated recurrent unit machine-learning algorithms. The purpose of this study is to demonstrate the ability of a physics-framed deep-learning-based approach to generate predictions of two-phase immiscible flow associated with CCUS at a tiny fraction of the time required by state-of-the-art numerical reservoir simulation tools. To accomplish this, we used the two-phase flow equation for immiscible displacement of water by CO_2 to frame our supervised deep-learning problem. This problem formulation technique is an alternative to the method presented by Raissi et al.[14, 15] for incorporating physics into neural network training. The neural network architecture for each algorithm was optimized through hyperparameter tuning and multiple methods were used to

thoroughly evaluate the performance of the models. Although the context for our demonstration is CCUS, the methods are broadly applicable to reservoir simulation in general.

2. METHODOLOGY

2.1 Background on Supervised Machine-Learning Methods

Multi-layer perceptron (MLP), convolutional neural network (CNN), long short-term memory (LSTM) cell, and gated recurrent unit (GRU) cell are among the commonly used deep-learning algorithms[16]. MLP is a class of feedforward neural networks with network architecture defined by its width (i.e., number of neurons or nodes per hidden layer) and depth (i.e., number of hidden layers). CNN performs a series of convolutional operations on the input data, each acting on a different slice of the input array, otherwise called the convolutional filter. LSTM and GRU, each comprising four and three built-in layers respectively, are special sub-classes of Recurrent Neural Network (RNN), capable of learning long-term dependencies in time-series data. LSTM and GRU feed time-series data through their respective sub-layers that interact in a specific way and are therefore able to learn temporal evolution of reservoir pressure, CO_2 saturation, well extraction rate, and bottomhole pressure. Other commonly used deep-learning frameworks are Autoencoders (AEs) and Generative Adversarial Networks (GANs). AEs are data compression and reconstruction techniques that are often built upon MLP, CNN, LSTM, and GRU algorithms with an architecture that primarily comprise an encoder and a decoder. GAN is a class of deep generative models that mainly consists of a generator and a discriminator with both adversarial networks closely connected to each other. Applied to subsurface flow problems, some of these algorithms have demonstrated remarkable capabilities to capture complex flow patterns among reservoir state variables[9, 11, 17-25].

Because neural network capacity is determined by network width and depth, deep neural networks are often needed to learn complex patterns in the data. However, the deeper or wider the network, the larger are its capacity and number of variables that must be learned during

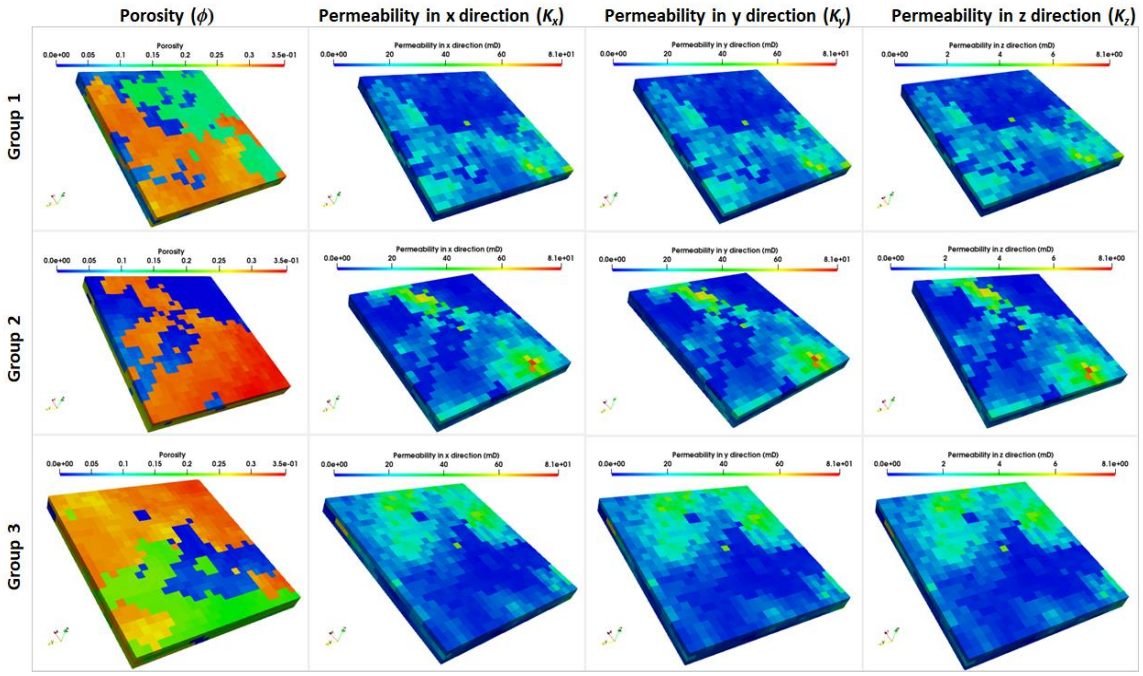
network training. Depending on the size of the data, large capacity neural nets can take a very long time to train and may require supercomputing resources. Typical large-scale CCUS reservoir models have on the order of hundreds of thousands to millions of grid blocks, thus arbitrary selection of network parameters could easily result in extremely large networks that will be very difficult to train on standard desktop computers. Besides, the intended purpose for increasing network size may not be realized; that is, a large capacity neural network may not necessarily improve the performance of the proxy model over a simple network. Even when performance is improved, there could be other approaches that avoid additional computational penalties that arise from complex neural network architectures. These constraints necessitate the need to optimize the size of neural networks. Furthermore, the physical laws governing two-phase flow in porous media are not naturally honored by off-the-shelf deep-learning algorithms such as MLP, CNN, LSTM, and GRU. To address the need to accurately model flow and transport in the reservoir, we have included the underlying physics of flow in our formulation of the deep-learning problem. Recent studies on other fluid-flow problems have focused on improving predictions made with these algorithms by including an additional term in the total loss function from the governing partial differential equation along with the neural network loss function[14, 15]. Using the MLP, CNN, LSTM, and GRU algorithms, our goal is to develop lean, efficient, and fast DL-based proxy models that reasonably represent the equivalent full-physics model for CO_2 displacement of water in a porous media. The studies referenced above typically use only the coordinate information of grid blocks along with time (i.e., x, y, z, and t) as inputs for the physics-informed neural network. Other studies on data-driven surrogate modeling use just the well injection rate, permeability and/or porosity fields, and time as inputs without considering the underlying physics of flow through porous media in the framing of their machine-learning problem[26, 27]. For subsurface fluid flow problems, however, rock properties, well constraints, and history of the state of the reservoir are critical for accurate forward modeling of the reservoir state variables. Therefore, the proxy models developed in this study use an exhaustive list of features such as heterogeneous

matrix porosity, heterogeneous diagonal permeability tensor, structured or unstructured mesh, constant well rates and bottomhole pressure, time-step size, and historical pressure and saturation data containing initial and boundary conditions. These features are then used in the proxy models to predict the temporal evolution of well flow rates and bottomhole pressure at both the injector and producer, and the spatio-temporal evolution of reservoir pressure and CO_2 saturation at all of the grid blocks in the domain.

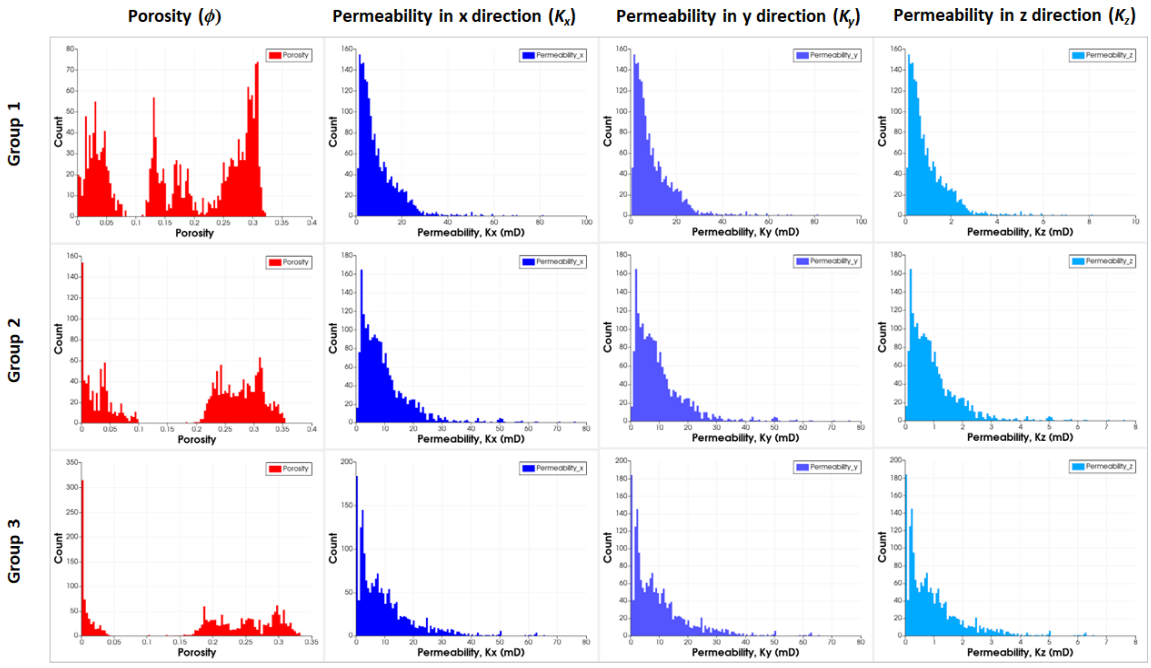
2.2 Data Generation using Full-Physics Numerical Simulations

The first step in data-driven deep-learning-based surrogate modeling is data generation. The data used in this study were generated via numerical simulation of a CO_2-water system under isothermal conditions using *CMG-GEM*, a fully compositional, general-purpose reservoir simulator[28]. The 3D geological model comprises twenty-seven realizations of matrix porosity and directional permeabilities (k_x, k_y, k_z) in 25 x 25 x 3 Cartesian grid blocks, with each grid block measuring 300 ft by 300 ft by 11.1 ft in x, y, and z directions respectively. The diagonal permeability tensor at each grid block is such that $k_x = k_y$ and $k_z = 0.1 * k_x$, and like porosity, varies among grid blocks in the x, y, and z directions. The 27 realizations of static models contain three groups of porosity and permeability fields (Fig. 1a). Histograms of the porosity and permeability distributions are shown in Fig. 1b. Porosity is generally multimodal while permeabilities are positive-skewed lognormal unimodal distributions. Two wells penetrate the three reservoir layers; one well is used to inject CO_2 and the second is used to extract water and is intended as a relief well for excessive pressure buildup in the reservoir. CO_2 is injected into the first well at 13 different constant rates that are spread over the three groups of porosity and permeability fields (Fig. 1c). These groupings, in addition to the grid-block-to-grid-block variations in porosity and permeability tensor, introduce spatial and temporal heterogeneities into the data, which are critical for developing a robust surrogate model. Water is extracted from the second well at constant bottomhole pressure (BHP) of 3525 psi. Full reservoir simulation was performed for 72 months

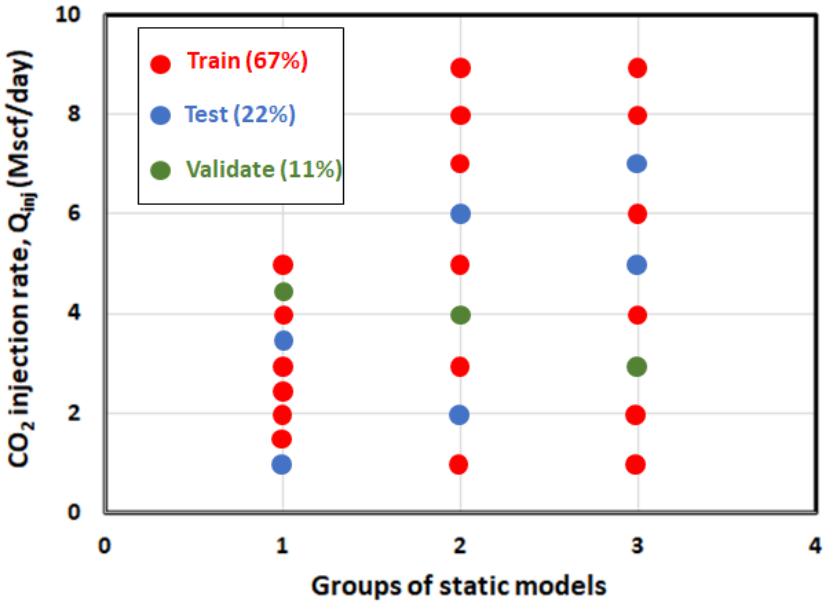
(2161 days in total) for each of the 27 porosity and permeability realizations. The reservoir pressure and CO_2 saturation within every grid block, well extraction rate at the producer (volumetric rate at the surface), and bottomhole pressure at the injector were then extracted to be used to evaluate the different machine-learning algorithms.



(a)



(b)



(c)

Fig. 1: Static models and bottomhole injection rates used to simulate CO_2 injection into water-saturated reservoir: (a) 3D contours showing three groups of porosity and permeability fields used to generate 27 realizations of static models; (b) histograms of the porosity and permeability fields; (c) rate-static model matrix showing bottomhole CO_2 injection rate schedule spread over the three groups of static models

described in (a). The color-coded circles show how the rate-static model matrix is used to distribute data into training, validation, and testing data sets for deep-learning-based surrogate modeling.

2.3 Development of Deep-learning-Based Proxy Models

Numerical solutions of a given problem are by design unique irrespective of the numerical schemes used to discretize the PDE. The main difference is where the system variables are computed – at corner points, grid centers, or irregularly-distributed points. The key reason why ML-based (i.e., data-driven) surrogate models are proposed for forward modeling is not to replace conventional numerical simulation, but rather to complement it by addressing the problem of using numerical codes during history matching. It is a well-known fact that forward modeling through numerical simulation is computationally expensive, especially when dealing with field-scale reservoir management (in some cases taking several days or weeks to complete one simulation run). This computational burden is exacerbated when numerical simulators have to be used repeatedly during inverse modeling. This is where "data-driven" models excel; they can be executed rapidly during history matching with excellent agreement with the numerically simulated data and are thus computationally efficient proxies for the numerical model. Unlike traditional full-physics models, supervised machine-learning algorithms can be used to develop surrogate models for predicting subsurface variables at near-real time, thus are well-suited for developing virtual learning environments for subsurface flow processes. At the heart of supervised learning with these algorithms is the estimation of the function that maps a given set of subsurface data (e.g., formation and fluid properties) to the reservoir state variables (e.g., pore pressure, CO_2 saturation, etc.). This mapping is described by Equation 1:

$$Y = f\left(X\right) \quad \ldots\ldots (1),$$

where Y represents the target variables (e.g., reservoir pore pressure, CO_2 saturation, etc.), X represents the features or inputs (e.g., porosity, permeability, etc.), and ***f*** represents the function that maps X to Y, which is learned during training.

2.3.1 Physics-Guided Formulation of Supervised Deep-learning

To formulate the deep-learning problem statement in this study, we begin by examining the partial differential equation governing fluid flow through porous media. Under zero capillary pressure, negligible gravity effect, and isothermal conditions, the transport equation for each phase in a CO_2-water system is given by:

$$\frac{\partial}{\partial t}\left(\phi\rho_\alpha S_\alpha\right)+\nabla\bullet\left(-\frac{\rho_\alpha\overline{\overline{\mathbf{K}}}k_{r\alpha}}{\mu_\alpha}\nabla p\right)=\rho_\alpha\bar{q}_\alpha;\quad \alpha=\mathrm{CO_2},\ \text{water} \qquad \text{..... (2),}$$

where,

$\phi=$ matrix porosity; $\rho_\alpha=$ fluid phase density; $S_\alpha=$ fluid phase saturation; $t=$ time;
$\overline{\overline{\mathbf{K}}}=$ permeability tensor; $k_{r\alpha}=$ relative permeability; $\mu_\alpha=$ fluid phase viscosity;
$\bar{q}_\alpha=$ volumetric fluid phase injection or extraction rate per unit volume

When expressed explicitly for each phase, Equation 2 is nonlinear because CO_2 density depends on the pore pressure, and relative permeability depends on saturation. Given the initial and boundary conditions, the pressure and saturation solutions of Equation 2 vary in space and time, i.e. $P,\ S_\alpha = \boldsymbol{f}(x, y, z, t)$. A very common strategy is to set P and S_α as targets for neural networks and x, y, z, and t as inputs for the networks. Although this may be sufficient for machine-learning algorithms to determine the mapping function $\boldsymbol{f}$, training the neural network in this manner would amount to a mere statistical exercise without regard to the unique correlations between flow properties and the underlying physics that govern subsurface fluid transport. Besides, the mapping function $\boldsymbol{f}$, during testing, may not generalize very well to never-before-seen subsurface data containing a different set of porosity and permeability fields. Another approach that was developed recently is to embed Equation 2 as an additional term in the loss function of the neural network, to be minimized during training[14, 15]. We propose an approach that uses the discretized form of Equation 2 as a basis to frame a physics-guided machine-learning problem. By following

the explicit and implicit formulations of the single- and two-phase forms of Equation 2, it is shown in the appendix that the time evolution of the state variables of interest can be simplified as:

$$p_{i,j,k}^{n+1} = f\left(p_{i,j,k}^{n},\ S_{i,j,k}^{n},\ \Delta x,\ \Delta y,\ \Delta z,\ \Delta t,\ \bar{q},\ \phi,\ k_x,\ k_y,\ k_z,\ c_r,\ \mu\right) \quad \text{(3a)}$$

$$S_{i,j,k}^{n+1} = f\left(S_{i,j,k}^{n},\ p_{i,j,k}^{n+1},\ \Delta x,\ \Delta y,\ \Delta z,\ \Delta t,\ \bar{q},\ \phi,\ k_x,\ k_y,\ k_z,\ c_r,\ \mu\right) \quad \text{(3b)}$$

Equation 3a means that the pore pressure ($p_{i,j,k}^{n+1}$) inside every grid block at the new time step depends on the grid block pressure ($p_{i,j,k}^{n}$) and saturation ($S_{i,j,k}^{n}$) at the old time step, grid-block size or grid information ($\Delta x, \Delta y, \Delta z$), time-step size ($\Delta t$), well constraints or inner boundary conditions ($\bar{q} = f\left(\text{well rate, well BHP}\right)$), and rock and fluid properties ($\phi,\ k_x,\ k_y,\ k_z,\ c_r,\ \mu$). Similarly, Equation 3b indicates that fluid saturation ($S_{i,j,k}^{n+1}$) inside every grid block at the new time steps depends on the grid-block pressure ($p_{i,j,k}^{n+1}$) at those time steps, grid-block pressure ($p_{i,j,k}^{n}$) and fluid saturation ($S_{i,j,k}^{n}$) at the old time steps, grid information, time-step size, well constraints or inner boundary conditions , and rock and fluid properties. Consequently, we could formulate our deep-learning problem statement as follows: given the dynamic variables (i.e., grid-block pressure, saturation, and surface extraction rate) at the old time steps, and the static variables (i.e., grid information, time-step size, well constraints or inner boundary conditions, and rock and fluid properties), we want to compute the grid-block pressure, saturation, and surface extraction rate at the new time steps. This formulation does not only allow the neural network to determine the mapping function ***f*** between the static variables and the dynamic state variables, it also ensures that the network learns the strong coupling between the state variables themselves (i.e., the dependence of pore pressure on saturation and vice versa), which is obvious from the nonlinearity of the governing fluid transport equation.

2.3.2 Data Preprocessing Prior to Network Training

Although we used a single time-step (Δt) approach in this study, our approach is neither restricted to single time steps nor is it limited to one-month time steps but can in fact accommodate any specified time-step size. There are multiple ways to formulate our time-series problem which includes: (i) given the available data at a single previous time t_n, predict results for future time t_{n+1}, where $t_{n+1} = t_n + \Delta t$. (ii) given the available data at a single previous time t_n, predict results for multiple future times $t_{n+1}, t_{n+2}, t_{n+3}$, etc. simultaneously; (iii) given the available data at multiple previous times $t_n, t_{n-1}, t_{n-2}, t_{n-3}$, etc., predict results for a single future time t_{n+1}; (iv) given the available data at multiple previous times $t_n, t_{n-1}, t_{n-2}, t_{n-3}$, etc., predict results for multiple future times $t_{n+1}, t_{n+2}, t_{n+3}$, etc. Exploring the performance of these different strategies is a topic for ongoing research.

For a ML-based surrogate model that is to be developed with pre-simulated data, all the above problem formulation strategies are theoretically possible. However, practical considerations in subsurface applications during early stages of field development narrow down the options to just strategies (i) and (ii). Strategies (iii) and (iv) are more appropriate during mid- to late stages in the life of a reservoir when significant well data are available. Strategy (i) is essentially the workflow in conventional reservoir simulation where pressure, saturation, rates, and bottomhole pressures are simulated beginning from the initial conditions (t_0) and matching forward in time steps (Δt) until the final simulation time is reached. Strategy (ii) does not match forward in time steps, but rather maps all future-times predictions to a single initial condition. While this may be convenient for fast forward modeling, it has two significant limitations. First the mapping of future-time predictions to a single initial condition disregards the intertwined mapping between the future-times predictions themselves which is clearly evident in Equations 3a and 3b; thus, this approach is more statistically driven than physics compliant. In contrast, strategy (i)

honors the interdependence of all future and previous time-step predictions as is always the case when simulations are matched going forward in time in conventional reservoir simulation. The second limitation is that strategy (ii) will increase the dimensionality of the machine learning problem, thus requiring needless large-capacity neural networks. For large reservoir models, this approach would require significant high-performance computing resources and very long training times. Therefore, our choice of strategy (i) is motivated by the desire to formulate a ML problem that does not just statistically fit the model features to the targets but does so in a manner that is scalable and consistent with standard practices in numerical reservoir simulation. Following Equations 3a and 3b, the static model and numerically simulated data from section 2.1 were split into static and dynamic data as illustrated in Fig. 2a. The dynamic data were split further into historical (i.e., old or previous time steps) and future-state (i.e., new or future time steps) data. The historical data were then combined with the static data and used as the features (i.e., input) while the future-state data were used as the target during network training. Specifically, p^{n+1}, S^{n+1}, and q^{n+1} were generated by splitting the numerically simulated data into previous time-step data (i.e., p^{n}, S^{n}, and q^{n}) and future time-step data (i.e., p^{n+1}, S^{n+1}, and q^{n+1}). After completing this preprocessing step, both sets of data were used to train the machine-learning models by using future time-step data as targets and previous time-step data combined with static data as inputs during model training. We also trained a separate auxiliary machine-learning model that predicts previous time-step data using only static variables and a series of previous time steps as input. A total of thirteen features and three targets is used for network training. Rock properties (i.e., matrix porosity and permeability), and by extension grid-block pressure and saturation, depend on grid-block volume; so rather than using (x, y, z) nodal coordinates as part of the input, we used (Δx, Δy, Δz) to form a Cartesian grid of the reservoir. For training, validation, and testing purposes, we further distributed the features and targets of all 27 static model realizations into fractions comprising 67%, 11%, and 22% of the total, respectively. Thus,

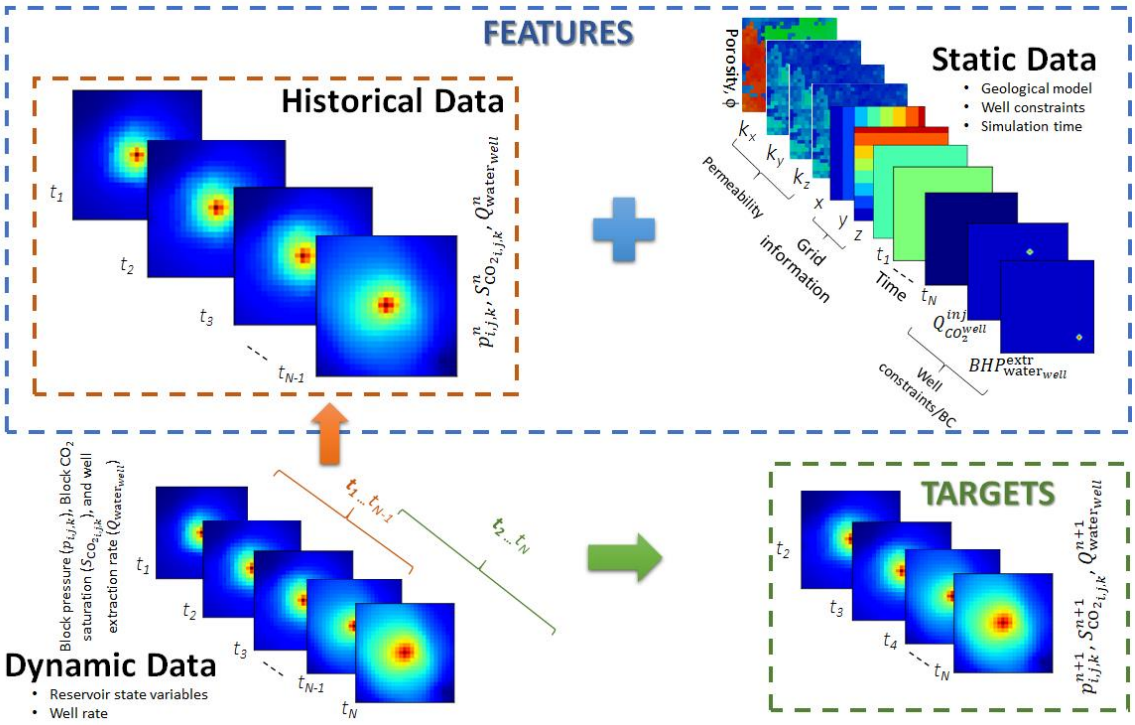
the features and targets from 18 different realizations were used for training, three realizations for validation, and six realizations for testing. As noted previously, 13 different constant bottomhole CO_2 injection rates were simulated across the three groups of porosity and permeability fields. As color-coded in Fig. 1c, data splitting for network training was carefully done by ensuring that the different groups of permeability fields and bottomhole injection rate are well-distributed within training, validation, and testing data sets. This is necessary because both permeability and bottomhole injection rate impact fluid flow and machine-learning algorithms tend to perform well when features are well-distributed in the data used to train the neural network, thereby ensuring broad sampling of the features space. Following standard practice, the validation data are used to optimize training of the neural network, while testing data are not used for training but are reserved to evaluate how the proxy models generalize to never-before-seen data.

2.3.3 Neural Network Algorithms, Architectures, and Training

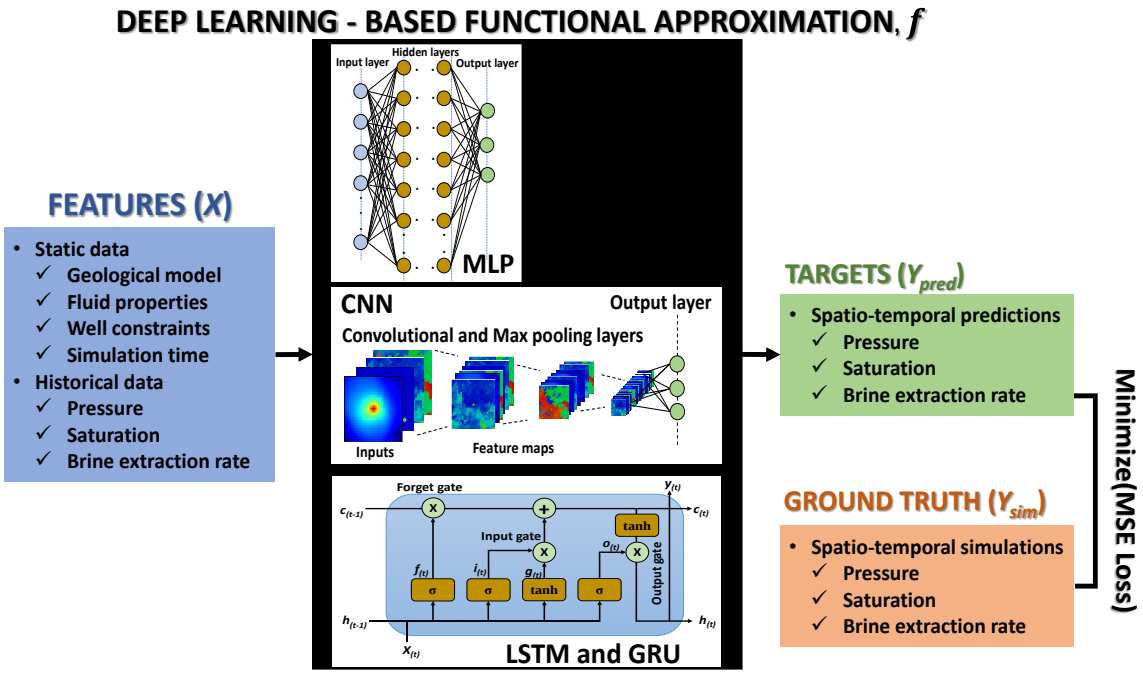
Four supervised neural network algorithms were evaluated in this study. They include the MLP, the CNN, the LSTM, and the GRU. Fig. 2b illustrates the network architecture, including the input and output parameters, of the multivariate input, and multivariate output surrogate models developed in this study. All network features and targets were normalized prior to training to speed up the learning process (faster convergence). Network hyperparameters of each algorithm were first optimized to determine the basic and leanest architecture with the most efficient performance. The neural net was implemented with the *Tensorflow* 2.1.0 library[29] and Python packages on an Intel® Core™ CPU @ 3.70GHz, 32.0 GB RAM. Optimization of the network loss function (Equation 4) is performed using the Adam optimization technique.

$$\text{Mean Squared Error (MSE) loss} = \frac{1}{N}\sum_{i=1}^{N}\left(Y_i^{\text{neural net}} - Y_i^{\text{simulation}}\right)^2;\ \text{where } Y = p_{i,j,k}^{n+1},\ S_{CO_{2_{i,j,k}}}^{n+1},\ q_{water_{well}}^{n+1} \quad \ldots\ldots (4)$$

We have intentionally avoided creating separate proxy models for the reservoir state variables considered in this study. Rather, we developed a single proxy model for each of the supervised deep-learning algorithms we examined, which can forecast all the state variables at once. One reason for adopting this approach is because multiphase flow problems are highly coupled. Although separate surrogate models are simpler to implement, doing so amounts to decoupling the critically important relationships among the state variables (as discussed in Section 2.2.1). For example, we know from multiphase flow that pore pressure and CO_2 saturation are strongly coupled through Equation 2. So, if we were to attempt to obtain the closed-form analytical solutions for the two-phase governing equations given a set of initial and boundary conditions, which would be an exercise in futility due to the strong nonlinearity of the partial differential equation, the pore pressure solution would depend on CO_2 saturation and vice versa. Besides, developing a separate proxy model for every state variable is an unwarranted duplication of efforts during training of the individual models and could impose additional penalties in training time, especially for large-scale subsurface reservoir models.


FEATURES
Historical Data
Static Data
Geological model
Well constraints
Simulation time
Porosity, ϕ
Permeability
Grid information
Time
Well constraints/BC
Dynamic Data
Reservoir state variables
Well rate
Block pressure, Block CO2 saturation, and well extraction rate
TARGETS

(a)


DEEP LEARNING - BASED FUNCTIONAL APPROXIMATION, f
FEATURES (X)
Static data
Geological model
Fluid properties
Well constraints
Simulation time
Historical data
Pressure
Saturation
Brine extraction rate
Input layer
Hidden layers
Output layer
MLP
CNN
Convolutional and Max pooling layers
Inputs
Feature maps
LSTM and GRU
Forget gate
Input gate
Output gate
TARGETS (Ypred)
Spatio-temporal predictions
Pressure
Saturation
Brine extraction rate
GROUND TRUTH (Ysim)
Spatio-temporal simulations
Pressure
Saturation
Brine extraction rate
Minimize(MSE Loss)

(b)

Fig. 2: Multivariate input, multivariate output deep-learning-based proxy modeling framework for subsurface carbon storage: (a) data preprocessing for network training and validation with snapshots of features and targets illustrating how static and dynamic subsurface data are used to train the neural networks; (b) network inputs, algorithms of interest, outputs, and label data used to minimize network loss (mean squared error or MSE) during model training.

3. RESULTS AND DISCUSSION

The models presented in this study have been tested with both 2D and 3D problems but for the sake of brevity and compactness, only results from the more general 3D data sets are discussed, because they demonstrate applicability of the proxy models to practical problems. First, we present results of hyperparameter tuning to determine the leanest network architectures that demonstrate acceptable performance. Then the performance of the surrogate models, including their compliance with the governing physical laws, is discussed using bivariate analysis (with concordance plots) and univariate analysis (with histograms of error distribution and assessment of 2D contours of reservoir pressure and CO_2 saturation in the vertical direction).

3.1 Hyperparameter Optimization

The size of a neural network, which is otherwise called network capacity, influences the performance of surrogate models, both in terms of training and validation as well as how the model generalizes to new data. Generally, neural network capacity is determined by the width (number of nodes, filters) and depths (number of hidden layers) of the network. The higher the width and/or depth, the higher the capacity of the network. A large-capacity network requires significantly large number of network parameters (i.e., weights and biases) to be optimized during training, which may lead to overfitting and may significantly increase training time, especially for large-scale problems such as those related to CCUS. Besides, the number of weights and biases will influence the storage requirement of the model after training[30]. For these reasons, it is inefficient to indiscriminately select network parameters for a given problem without first determining the smallest size of the network that would give approximately the same, if not better,

performance than large-capacity network architectures. Rather than arbitrarily selecting the number of hidden layers, number of nodes per layer, number of filters, and the learning rate (a hyperparameter that controls how much to change the model in response to the estimated error each time the model weights are updated), we first conducted training experiments with these hyperparameters to determine the minimum requirement for optimum model performance. Table 1 shows the effect of MLP, CNN, LSTM, and GRU hyperparameters on network loss after 100 epochs.

As the learning rate varied, network loss for each of the four algorithms decreased initially followed by an increase. In all cases, the least network loss occurs at a learning rate of 10^{-2}. Therefore, a learning rate of 10^{-2} is selected as the optimum value and is used for all subsequent training. The least network loss for MLP and CNN was found to occur at 100 nodes after experimenting with three different numbers of nodes per hidden layer. For the CNN, this value represents the optimum number of nodes for the dense layer that succeeds the convolutional layer just prior to the output layer of the network. For the LSTM and GRU, 50 nodes in the hidden layer give the least network loss. Convolutional filters in a CNN layer are used to extract features from the input data. Higher number of filters may be needed for complicated problems, but an excessive number of filters increases network capacity, which in turn increases training time. In our case, 64 convolutional filters appear to give the least network loss. However, the network loss with 32 convolutional filters is practically the same, so we will use this smaller number in our subsequent analysis to keep training time per epoch to the barest minimum. Finally, four hidden layers give the least network loss for the MLP and CNN algorithms. We stopped short of conducting the same analysis for LSTM and GRU because these algorithms are inherently designed with four and three sublayers respectively, thus the number of training parameters for a single layer of these time-series algorithms is equivalent to those of a four-layer MLP and CNN. Furthermore, increasing the number of hidden layers would result in network capacity that would be difficult to tune on a desktop computer.

Table 1: Network loss after 100 epochs of hyperparameter optimization

Hyperparameter space		Deep-learning algorithm			
		MLP	CNN	LSTM	GRU
Learning rate*	$1.00x10^{-1}$	$1.56x10^{-2}$	$9.61x10^{-3}$	$1.59x10^{-4}$	$4.25x10^{-4}$
	$1.00x10^{-2}$	$1.28x10^{-4}$	$1.10x10^{-4}$	$1.13x10^{-4}$	$1.00x10^{-4}$
	$1.00x10^{-3}$	$3.77x10^{-4}$	$1.89x10^{-4}$	$3.62x10^{-3}$	$1.81x10^{-3}$
Number of nodes per hidden layer**	20	$1.47x10^{-3}$	$1.75x10^{-4}$	$2.60x10^{-4}$	$1.11x10^{-4}$
	50	$3.70x10^{-4}$	$1.13x10^{-4}$	$1.07x10^{-4}$	$1.01x10^{-4}$
	100	$1.79x10^{-4}$	$8.18x10^{-5}$	$1.52x10^{-4}$	$1.01x10^{-4}$
Number of filters†	16	N/A	$1.14x10^{-4}$	N/A	N/A
	32	N/A	$6.77x10^{-5}$	N/A	N/A
	64	N/A	$3.86x10^{-5}$	N/A	N/A
Number of hidden layers††	1	$1.37x10^{-4}$	$1.16x10^{-4}$	√	√
	2	$1.15x10^{-4}$	$9.62x10^{-5}$	x	x
	4	$9.58x10^{-5}$	$9.47x10^{-5}$	x	x

* number of nodes per hidden layer, number of filters (CNN only), and number of hidden layers were fixed at 100, 32, and 1 respectively

** learning rate, number of filters (CNN only), and number of hidden layers were fixed at $1.00x10^{-2}$, 32, and 1 respectively

† learning rate, number of nodes per hidden layer, and number of hidden layers were fixed at $1.00x10^{-2}$, 100, and 1 respectively

†† learning rate, number of nodes per hidden layer, and number of filters (CNN only) were fixed at $1.00x10^{-2}$, 100, and 32 respectively

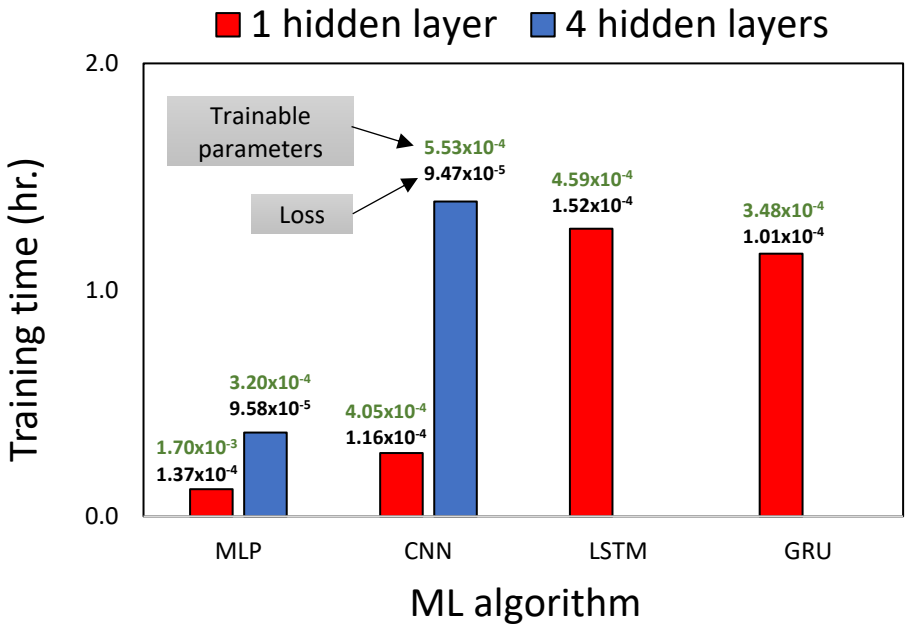
N/A not applicable

√ hyperparameter space was sampled

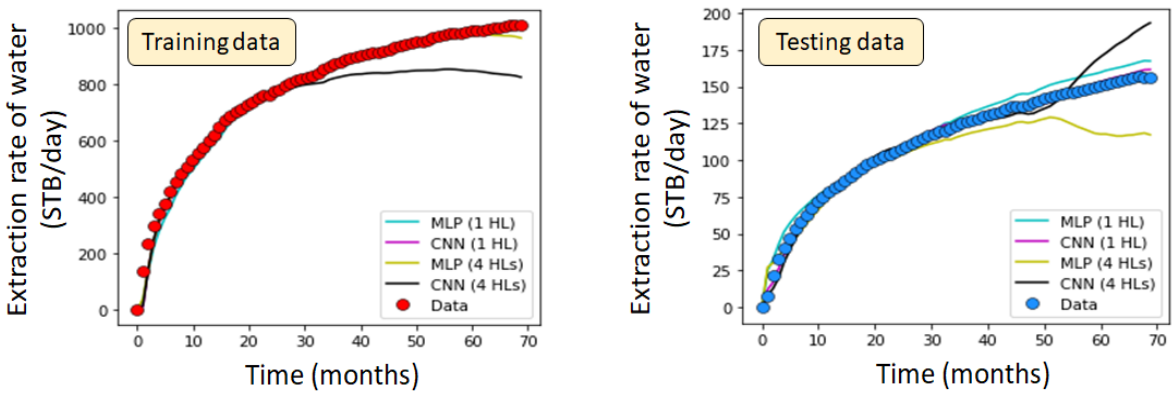
x hyperparameter space was not sampled

Fig. 3a shows the effect of increasing network layers on the training time for all four algorithms. The number of trainable parameters, which is an indication of the network size, is displayed on the plot along with the corresponding network loss from Table 1. This shows that the training time increases with the network capacity. For example, the training time for the MLP

algorithm increases by nearly three times after the network hidden layer is quadrupled, without any significant improvement in network loss. This indicates that a single layer MLP will perform equally well as a four-layer MLP, without imposing unwarranted training cost. A similar result was obtained for the CNN where training time increased by four-fold after increasing the number of hidden layers from one to four. As noted previously, results for one layer of LSTM and GRU are equivalent in this case to those of the four-layer CNN. Fig. 3b shows water extraction rate (volume corresponding to surface conditions) from one realization of the training and testing data sets after prediction with one- and four-layer MLP and CNN proxy models. For the training data, one-layer MLP and CNN matched the data but the four-layer models, especially CNN, did not capture the late-time trend. Similar behavior can be seen in the testing data. Therefore, adding hidden layers without justification may not necessarily improve model performance. Instead the model may become overfitted and will learn trends that do not generalize to the entire data sets. A simple single-layer network architecture, developed for a carefully framed machine-learning problem, is sufficient in our case and is used going forward in the sections that follow. This underscores the necessity for hyperparameter optimization to reduce training time and avoid incurring additional computational penalties.



(a)



(b)

Fig. 3. Impact of neural network layers on: (a) training time; (b) model prediction of water extraction rate (surface conditions) at the producer. HL represents a hidden layer.

3.2 Model Training and Performance Metrics

Fig. 4 shows the status and performance of the network as training progresses. First, we present the error distribution for each of the target variables as predicted by the MLP algorithm after just 10 epochs (Fig. 4a). These distributions are Gaussian with initially high standard deviation (σ)

and mean (μ). Later, we will present the error state after the training is completed. Evolution of the combined estimate of the network mean squared error (MSE) for all three target variables as the training progresses (Fig. 4b) shows that the MSE for the MLP model is significantly minimized reaching approximate final values of 8.56x10^{-6} and 9.45x10^{-6} for the training and validation sets respectively. We observe the closeness between the MSEs of both data sets at every epoch, an indication that the network is well-trained to generalize to not just the training data, but also to new sets of data. This indicates that the proxy model has not been influenced by overfitting.

Because the network MSE only monitors the error between predicted variables and their true values, we also computed MSE on the finite-difference discretized global residual function of the two-phase flow partial differential equation defined by Equation 5 to determine whether the ML model-predicted variables honor the underlying physical laws governing two-phase flow. These results are presented in the left-hand-side graph of Fig. 4c. The final MSEs are 1.59x10^{-11} and 1.956x10^{-11} for the training and validation sets, respectively.

$$f = \begin{bmatrix} f_{\mathrm{CO_2}} \\ f_{\mathrm{water}} \end{bmatrix} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (5),$$

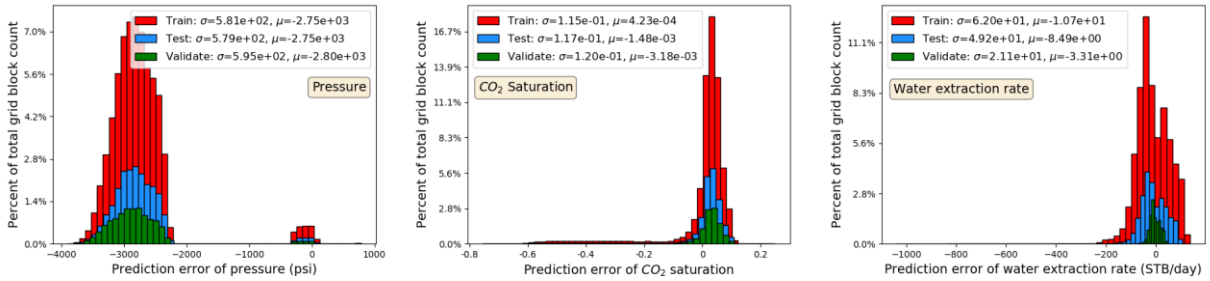
where,

$$f_{\mathrm{CO_2}} = \phi \frac{\partial S_{\mathrm{CO_2}}}{\partial t} + \nabla \bullet \left(-\frac{\overline{\overline{\mathbf{K}}} k_{r\mathrm{CO_2}}}{\mu_{\mathrm{CO_2}}} \nabla p \right) - \bar{q}_{\mathrm{CO_2}}$$

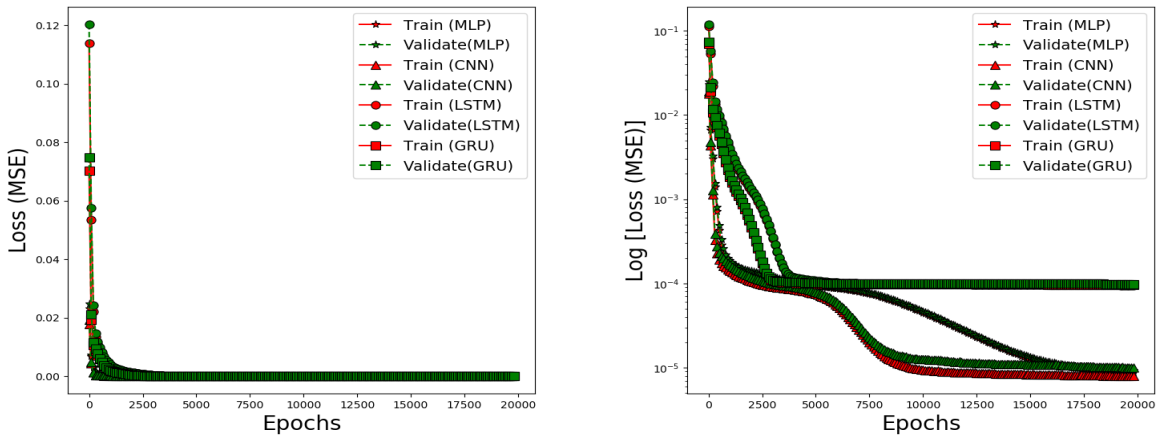
$$f_{\mathrm{water}} = \phi \frac{\partial S_w}{\partial t} + \nabla \bullet \left(-\frac{\overline{\overline{\mathbf{K}}} k_{rw}}{\mu_w} \nabla p \right) - \bar{q}_w$$

Recognizing on one hand that the input and target variables of the neural network were normalized prior to network training, this being a necessary preprocessing step for features and targets, and on the other hand that Equation 5 expresses dimensional variables, it is important to verify that these MSEs truly reflect model performance. To address this, we reverted all ML-

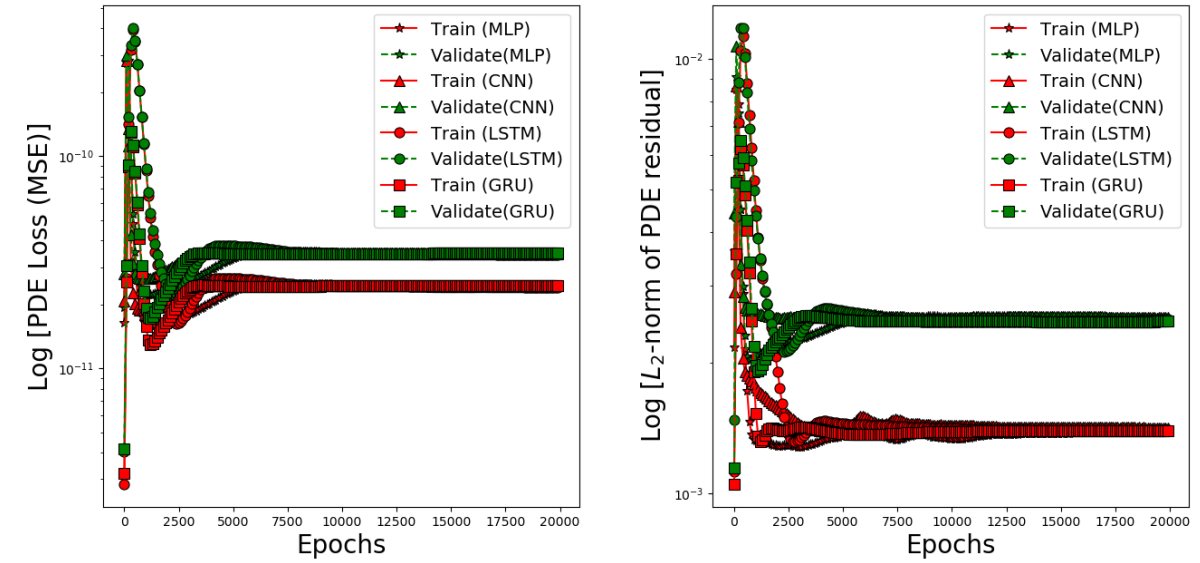
predicted normalized variables to their dimensional states and then used these values to compute the L_2-norm of the global residual function defined by Equation 5. The right-hand-side graph in Fig. 4c shows the outcome for one of the static model realizations at one time step. The final values of the L_2-norm are 1.28×10^{-3} and 2.02×10^{-3} for the training and validation sets respectively. In Figs. 4(d) through 4(e), we present contour plots to show how the final L_2-norms computed with ML-predicted pressure, CO_2 saturation, and water extraction rate (surface conditions) for all static models at all simulation times compare with the L_2-norms computed with the corresponding numerically simulated variables. As expected from a converged numerical solution, these values are approximately zero and compare reasonably well. This demonstrates that the surrogate model has learned the important physics-constrained fluid flow dynamics and is a critical verification for our methodology and results.



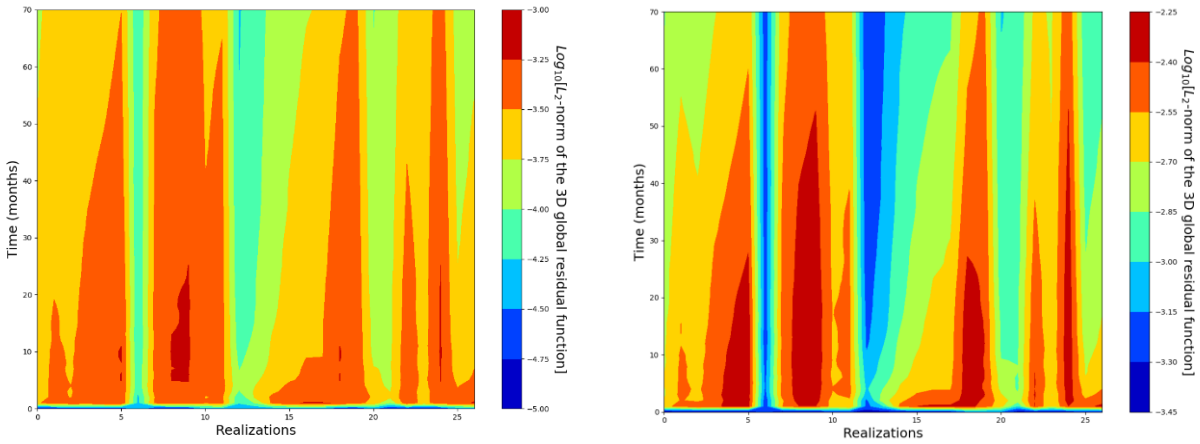
(a)



(b)



(c)



(d) (e)

Fig. 4 Network status and performance during training: (a) grid-block-by-grid-block error distribution after 10 epochs of training with MLP algorithm (2.40×10^6, 3.99×10^5, and 7.99×10^5 data points are displayed for training, validation, and testing, respectively; standard deviation (σ) and mean error (μ) are reported in the same units as the x-axes); (b) cartesian (left-hand side) and log-scale (right-hand side) plots of the evolution of network loss (mean squared error or MSE) during network training; (c) MSE of the two-phase flow PDE global residual function (left-hand side) and L_2-norm (right-hand side) after Equation 5 is computed with surrogate model predictions using the final time-step data of one randomly selected static model; (d) contours of the L_2-norm computed with the original numerically simulated variables; (e) contours of the final L_2-norm computed with ML-predicted target variables. (d) and (e) were generated

with data sets from all simulation times in the 27 static model realizations. The MLP surrogate model was used to generate (e).

Table 2 summarizes the key network performance indicators, including the global MSE losses of the neural network, estimated model accuracy, training time per epoch, and prediction time for one static model realization comprising all features and target variables at all simulation times. Training losses of the four algorithms compare favorably with validation and testing losses. Estimated model accuracies are also equivalent and above 95%. The fact that the surrogate models perform equally well on the training, validation, and testing data sets indicates that overfitting is minimized. Typically, the difference between training and testing accuracies represents overfitting. It usually occurs when the performance of deep-learning proxy models on new, previously unseen data is significantly inferior to the performance on the training data. An overfitted proxy model learns the details in the training data to such an extent that it adversely impacts its performance on new data. As shown in Table 2, the surrogate models generally predict field pressure and CO_2 saturation plumes, in addition to the surface extraction rate of water, at all the simulation times for a given static model in less than seven seconds, with the MLP surrogate model demonstrating the best performance by balancing training speed, prediction time, and prediction accuracy with lean network capacity.

Table 3 shows how the prediction times in Table 2 for each surrogate model compares with CMG simulation run time. Predictions with the MLP, CNN, LSTM, and GRU models are about 16, 13, 11, and 11 times faster than the average CMG computational time for one blank test realization comprising 72 monthly time steps (i.e., 6 years) of data, respectively. It is worth noting that the total cost of developing an ML model comprises the data acquisition cost, data processing cost, programming cost, training cost, and computational (i.e., prediction/forecast) cost. For numerical simulation, the corresponding total cost includes the data acquisition cost (e.g., from seismic, drilling and coring, completions, laboratory PVT analysis, etc.), data processing cost, numerical simulator programming and debugging cost, numerical simulator installation/setup,

494 calibration, and support cost, and computational (i.e., simulation) cost. Because it is not a
495 standard practice when estimating the computational cost for numerical simulation to include all
496 the time it took to acquire and preprocess the relevant data needed as inputs to run reservoir
497 simulators on a computer and the time taken for numerical code development and testing, the
498 appropriate apple-to-apple comparison is to report the time it takes by the computer to execute
499 an already-developed numerical code (in the case of numerical simulators) and a pretrained ML
500 model to produce practically comparable values for the variables of interest throughout the entire
501 spatio-temporal domain; thus, only the ML model prediction times were used to estimate the ML
502 model speedups that are presented in Table 3. Even the ML model coding and training time is
503 unwarranted, although we provided this information in Table 2, else one would have to add the
504 cost of programming, debugging, installing, and supporting numerical simulators to the computer
505 run time when estimating the cost of numerical simulation. Obviously, this is not the case because
506 only the cost of executing numerical codes that are already developed and tested (e.g., CMG,
507 Eclipse, TOUGH, etc.) on a computer is usually reported.

508 **Table 2:** Performance metrics after network training

Data sets	**Metrics**	**MLP**	**CNN**	**LSTM**	**GRU**
Model Training					
Training (18 realizations)	Data points	2,396,250	2,396,250	2,396,250	2,396,250
	Learned Parameters	1,703	4,051	12,953	9,903
	Epochs	20,000	20,000	20,000	20,000
	MSE	8.77×10^{-6}	7.89×10^{-6}	9.46×10^{-5}	9.65×10^{-5}
	Accuracy (%)	97.27	96.78	97.75	97.26
	Time to train one epoch (secs.)	0.9	2.68	4.33	4.10
Model Predictions					
Validation	Data points	399,375	399,375	399,375	399,375

(3 realizations)	MSE	9.26×10^{-6}	1.00×10^{-5}	9.65×10^{-5}	9.85×10^{-5}
	Accuracy (%)	97.27	96.82	97.76	97.26
	Time for combined prediction of all variables at all simulation times in one realization (secs.) *	4.41	5.58	6.26	6.17
Testing (6 realizations)	Data points	798,750	798,750	798,750	798,750
	MSE	1.00×10^{-5}	8.92×10^{-6}	9.70×10^{-5}	9.91×10^{-5}
	Accuracy (%)	97.27	96.77	97.75	97.25
	Time for combined prediction of all variables at all simulation times in one realization (secs.) *	4.37	5.61	6.49	6.65

* These times represent the average length of time taken for combined prediction of reservoir pressure, CO_2 saturation, and water extraction rate at all simulation times in a given static model realization (computer configuration: Intel® Core™ CPU @ 3.70GHz, 32.0 GB RAM).

509 **Table 3.** Computational speedup of ML-based surrogated models over CMG simulation.

CMG numerical simulation			
Total elapsed time$^{\downarrow}$ (secs.)	**Minimum elapsed time (secs.)**	**Maximum elapsed time (secs.)**	**Mean elapsed time (secs.)**
1932.47	14.84	169.70	71.57*
ML model approximate speedup*			
MLP	**CNN**	**LSTM**	**GRU**
16x	13x	11x	11x
* 1 blank test realization comprising 72 monthly time steps (6 years) of data. $^{\downarrow}$ 27 realizations comprising 72 monthly time steps of data per realization.			

3.3 Model Prediction and Forecast

Machine learning workflow generally comprises of training and testing of models. Framing of the machine learning problem using the proposed methodology is required only during model training. For model testing (i.e., prediction and forecast), we avoided the need for simulation results at the previous time step by developing a separate auxiliary surrogate model to be used for predicting a series of previous time-step values using only static formation properties, constant flow rate, and a series of previous time steps as inputs. These were then used, in combination with the formation and fluid properties and a series of future time steps, to make future predictions and forecasts. To be specific, p^n, S^n, and q^n were first predicted using static variables and a series of previous simulation times t_n as inputs in the auxiliary model. The values of p^{n+1}, S^{n+1}, and q^{n+1} were then predicted by using p^n, S^n, and q^n combined with the static variables and a series of future simulation times t_{n+1}, where $t_{n+1} = t_n + \Delta t$, as inputs in the main model.

Thus, model predictions were performed in a single prediction step (i.e., all at once, not sequentially) given a set of input variables that includes a series of desired output times **(t_1, t_2, t_3,…, t_n)** that were separated by time step **Δt**: that is**, t_1, $t_1 + \Delta t$, $t_1 + 2*\Delta t$,…, t_n,** where **Δt** = 1 month for the datasets used in this study. It should be noted that **Δt** is not limited to one month but can in fact accommodate any specified time-step size and can vary from one simulation time to the next. Thus, in one prediction step, the ML models output 3D reservoir pressure, 3D CO_2 saturation, well extraction rate, and well injection *BHP* for all the 72 months in one-month intervals (i.e., one-month time steps).

The results of models trained with the complete 72 months of training data (please refer to Fig. 1c for data splitting) are discussed in sections 3.3.1 through 3.3.3. To demonstrate the proposed methodology in terms of accurate forecasting, we retrained the MLP model with only the first 36 months of training data (along with the first 45 months of validation data) and predict

using 72 months of blind test data comprising both the first and last 36 months of the blind test data (please refer to Fig. 1c for data splitting). The results are discussed in section 3.3.4.

3.3.1 CO_2 Bottomhole Pressure at the Injector

Because the scenario used in this study involves a constant-rate CO_2 injector along with a constant-BHP water producer, the BHP at the CO_2 injection well and surface water extraction rate at the producer must be estimated. Consequently, we included the water extraction rate (volumetric, at surface conditions) at the producer in our list of target variables for the neural network as noted previously. Rather than add BHP at the CO_2 injection well to this list for the neural network to learn, instead we retrieve the learned grid-block pressures of the grid blocks that host the CO_2 injection well and then we apply the standard Peaceman-type well model, Equation 5, to estimate BHP at the CO_2 injection well. The result of this calculation is compared to the actual numerically simulated BHP at the injector and is intended as a verification step for the proxy models. A well-matched BHP would mean that the coupling between wellbore hydraulics, otherwise called the inflow-performance relationship, and reservoir flow has been learned properly, because the well grid-block pressure is predicted using the surrogate model. The Peacemen model equations estimate bottomhole pressure as follows:

$$p_{bhp,\mathrm{CO}_2^{\mathrm{well}}}^{n+1,\ \mathrm{predicted}} = p_{i,j,k,\mathrm{CO}_2^{\mathrm{well}}}^{n+1,\ \mathrm{predicted}} + \frac{q_{\mathrm{CO}_2^{\mathrm{well}}}^{\mathrm{inj}}}{J_{\mathrm{CO}_2^{\mathrm{well}}}^{\mathrm{Peaceman}}} \quad \text{(6)},$$

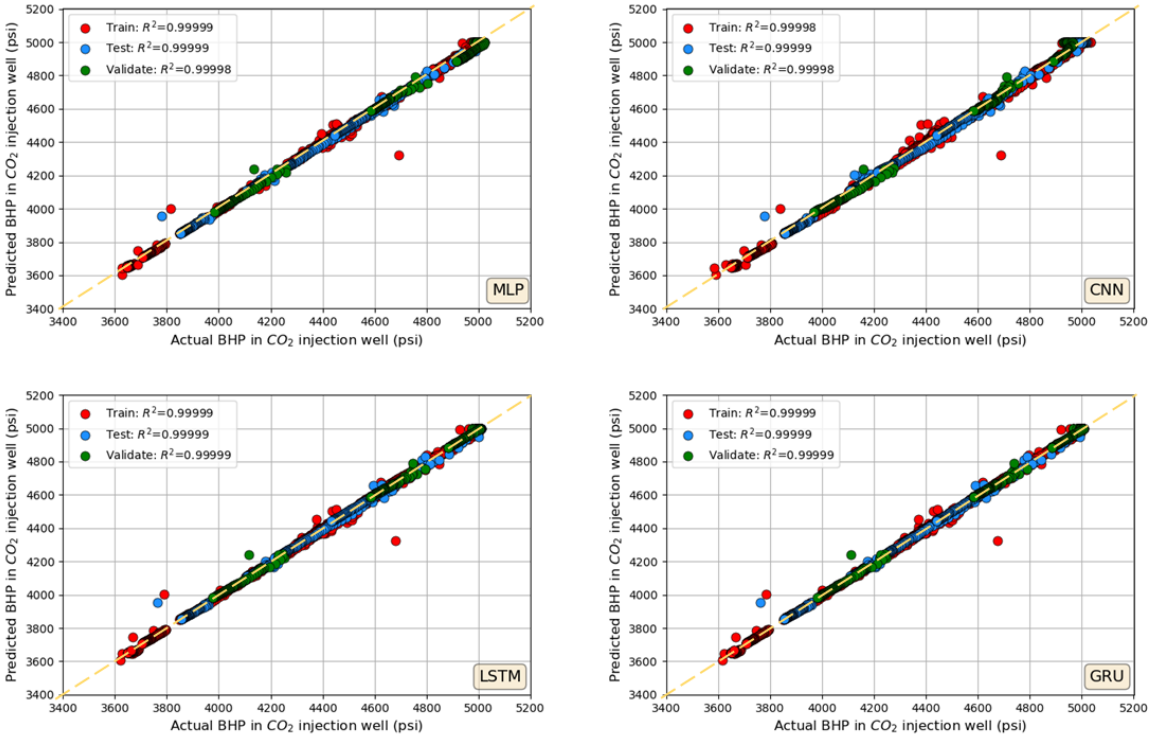
Where $q_{\mathrm{CO}_2^{\mathrm{well}}}^{\mathrm{inj}}$ is the constant bottomhole CO_2 injection rate and $J_{\mathrm{CO}_2^{\mathrm{well}}}^{\mathrm{Peaceman}}$ is the Peaceman CO_2 well injectivity index based on two-point flux approximation, defined as:

$$J_{\mathrm{CO}_2^{\mathrm{well}}}^{\mathrm{Peaceman}} = \frac{2\pi\left(h\sqrt{k_x k_y}\right)_{\mathrm{CO}_2^{\mathrm{well}}}}{\mu_{\mathrm{CO}_2}\log_e\left(\frac{r_{eq}^{\mathrm{CO}_2^{\mathrm{well}}}}{r_{well}^{\mathrm{CO}_2}}\right)}; \qquad r_{eq}^{\mathrm{CO}_2^{\mathrm{well}}} = \frac{0.28\left(dx^2\sqrt{\frac{k_y}{k_x}} + dy^2\sqrt{\frac{k_x}{k_y}}\right)^{0.5}}{\sqrt[4]{\frac{k_y}{k_x}} + \sqrt[4]{\frac{k_x}{k_y}}}$$

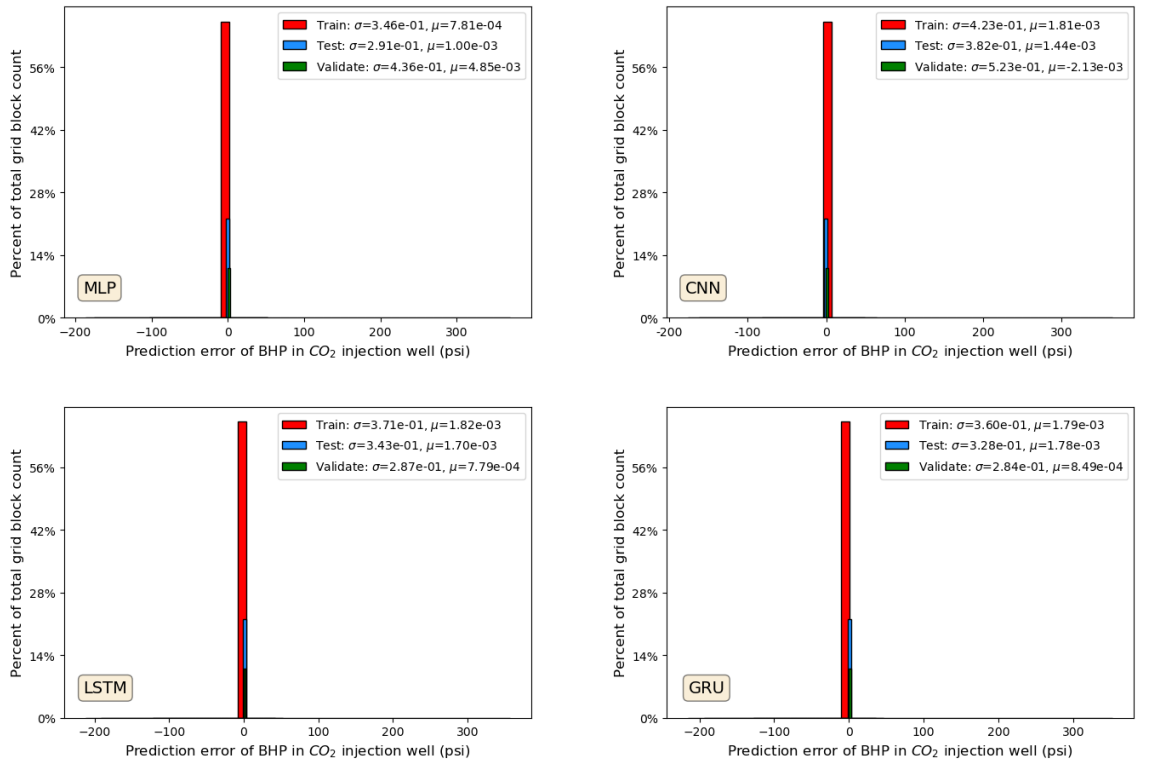
554 Recognizing that the Peaceman well model is strictly defined for two-point flux
555 approximation schemes such as commonly used in standard finite difference methods, and that
556 the numerical schemes implemented in commercial codes such as *CMG-GEM* may differ from
557 these simple schemes, thus potentially leading to variation in the actual $J_{\mathrm{CO_2^{well}}}^{\mathrm{Peaceman}}$ at different time
558 steps, we first computed the CO_2 well injectivity index using the above expression and then
559 attempted verification with equivalent estimates from the actual *CMG-GEM* data using Equation
560 6.

561 $$J_{\mathrm{CO_2^{well}}}^{\mathrm{CMG}} = \frac{q_{\mathrm{CO_2^{well}}}^{\mathrm{inj}}}{p_{bhp,\mathrm{CO_2^{well}}}^{n+1,\ \mathrm{simulated}} - p_{i,j,k,\mathrm{CO_2^{well}}}^{n+1,\ \mathrm{simulated}}} \quad \text{..... (7)}$$

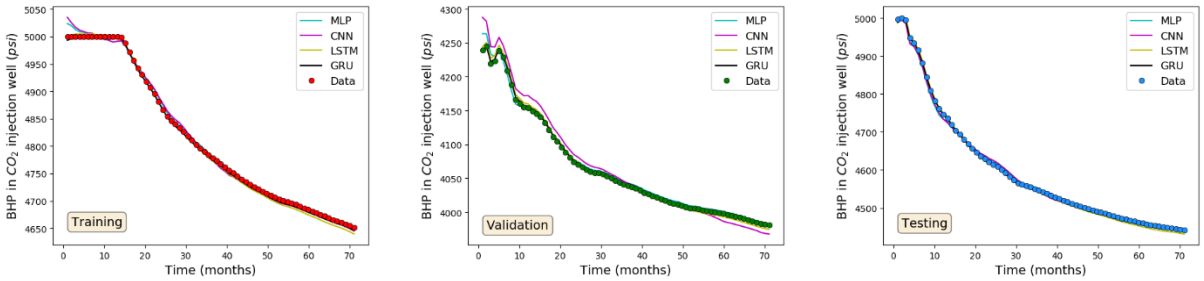
562 While $J_{\mathrm{CO_2^{well}}}^{\mathrm{Peaceman}}$ is not expected to vary with time following the above expressions, $J_{\mathrm{CO_2^{well}}}^{\mathrm{CMG}}$
563 actually varies with time. Thus, we used $J_{\mathrm{CO_2^{well}}}^{\mathrm{CMG}}$ (instead of $J_{\mathrm{CO_2^{well}}}^{\mathrm{Peaceman}}$) in Equation 5 to compute the
564 bottomhole pressure. Fig. 5 shows how predictions by the surrogate models compare to the
565 original data. Concordance plots for ML-predicted BHP and *CMG-GEM* output show excellent
566 agreement at all simulation times (Fig. 5a). This is supported by the R^2 coefficient. Error
567 distribution for the BHP also shows that predictions by the surrogate models are near-perfect (Fig.
568 5b). Standard deviations (σ) and mean errors (μ) of the different data sets are very low, thus
569 indicating that the surrogate models generalize to the entire data sets. Test cases of BHP vs.
570 time, one each for training, validation, and testing, show how well the predictions match the
571 original data (Fig. 5c). These plots confirm the results in the concordance plots and the error
572 distribution.


Train: R^2=0.99999
Test: R^2=0.99999
Validate: R^2=0.99998
MLP
Train: R^2=0.99998
Test: R^2=0.99999
Validate: R^2=0.99998
CNN
Train: R^2=0.99999
Test: R^2=0.99999
Validate: R^2=0.99999
LSTM
Train: R^2=0.99999
Test: R^2=0.99999
Validate: R^2=0.99999
GRU
Predicted BHP in CO_2 injection well (psi)
Actual BHP in CO_2 injection well (psi)

(a)


Train: σ=3.46e-01, μ=7.81e-04
Test: σ=2.91e-01, μ=1.00e-03
Validate: σ=4.36e-01, μ=4.85e-03
MLP
Train: σ=4.23e-01, μ=1.81e-03
Test: σ=3.82e-01, μ=1.44e-03
Validate: σ=5.23e-01, μ=-2.13e-03
CNN
Train: σ=3.71e-01, μ=1.82e-03
Test: σ=3.43e-01, μ=1.70e-03
Validate: σ=2.87e-01, μ=7.79e-04
LSTM
Train: σ=3.60e-01, μ=1.79e-03
Test: σ=3.28e-01, μ=1.78e-03
Validate: σ=2.84e-01, μ=8.49e-04
GRU
Percent of total grid block count
Prediction error of BHP in CO_2 injection well (psi)

(b)

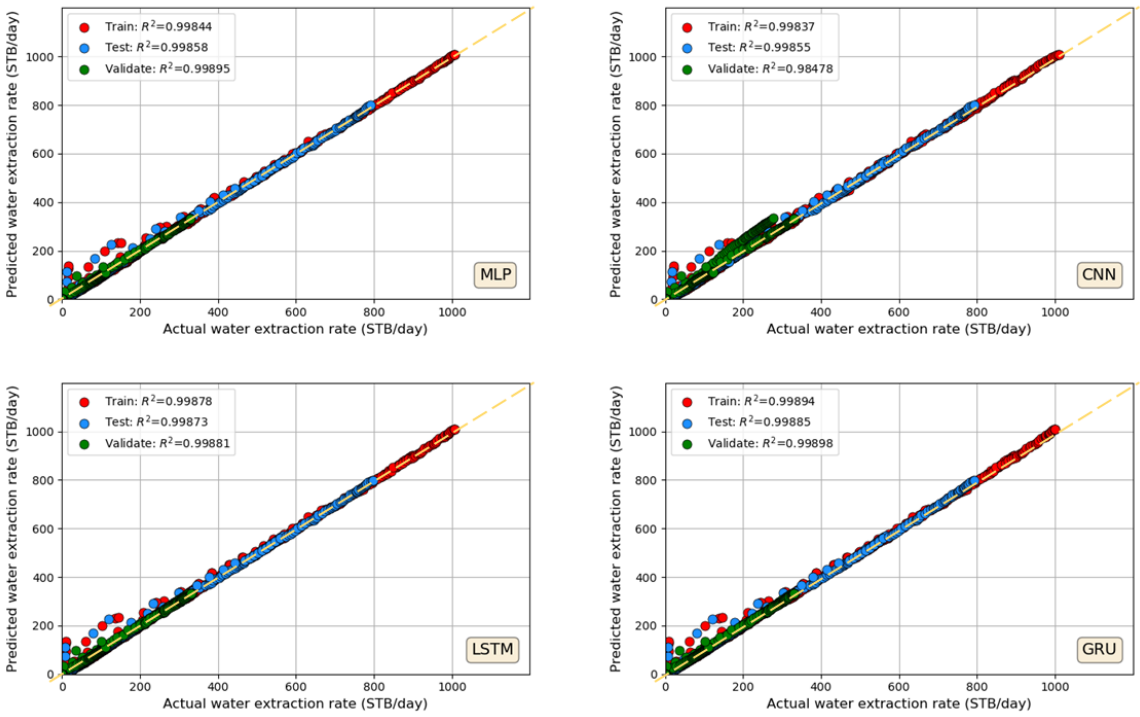


(c)

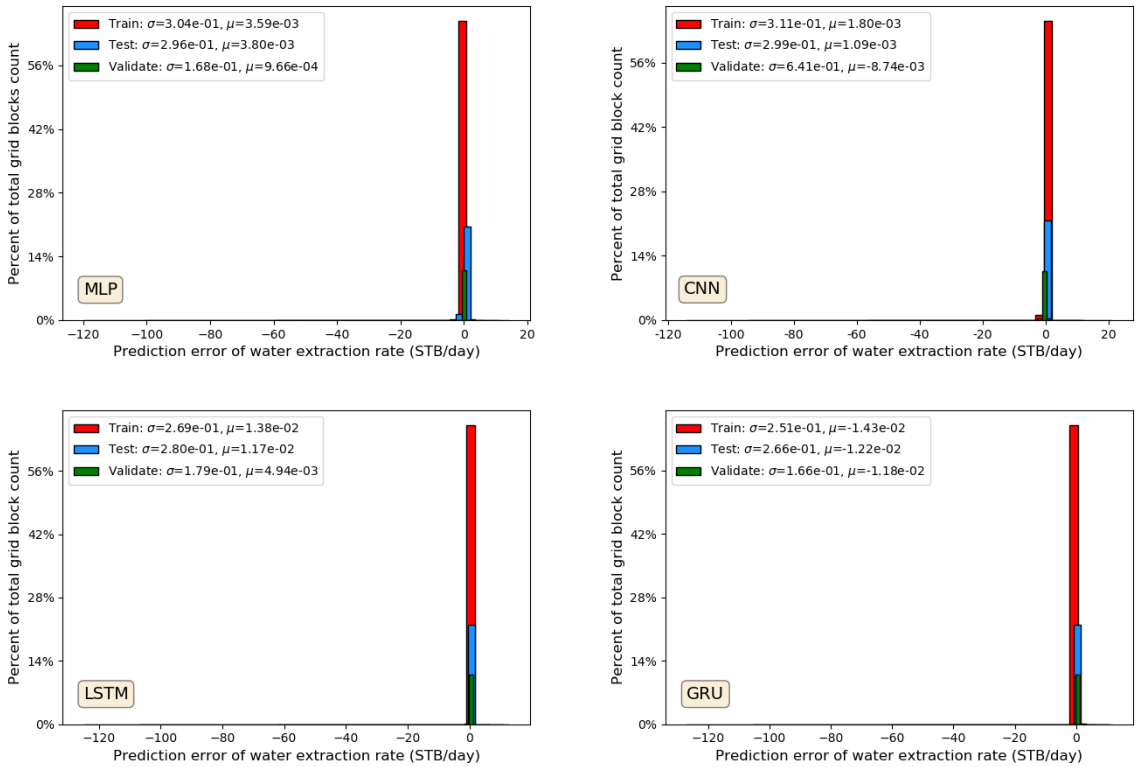
Fig. 5. Comparisons between the original data and predictions of injection well BHP by the four surrogate models: (a) grid-block-by-grid-block concordance plots at all simulation times and at injection well locations showing R^2 coefficients; (b) grid-block-by-grid-block error distribution showing the standard deviations (σ) and mean errors (μ); (c) sample BHP vs. time plots for each data set. The total number of data points shown in (a) and (b) comprises 2.40×10^6, 3.99×10^5, and 7.99×10^5 for training, validation, and testing, respectively.

3.3.2 Water Extraction Rate at the Producer

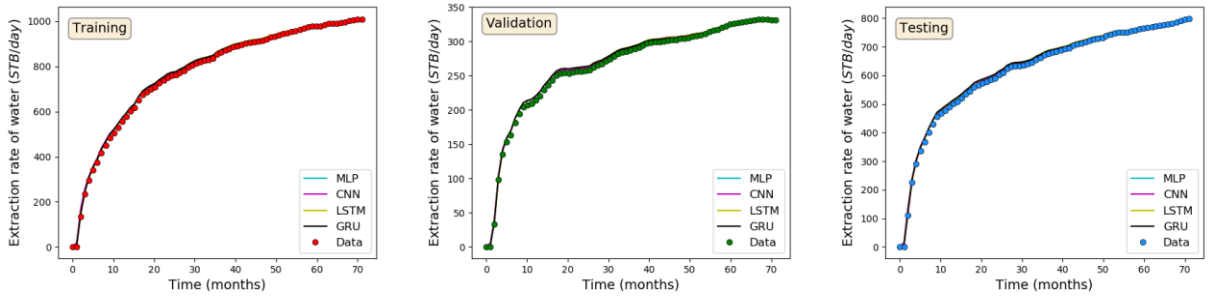
Fig. 6 shows comparison between the predicted water extraction rate (volumetric, at surface conditions) and the original *CMG-GEM* results. As shown in the concordance plot, the vast majority of the data points lie along the unit-slope line and the R^2 values indicate that water extraction rates predicted by the surrogate models reasonably match the labeled data at all simulation times. In addition, the error distributions approach zero for all four algorithms. This is supported by the standard deviations and mean errors which are very close to zero, thus indicating that the actual errors are near the mean error estimates. Compared to the error distribution of the water extraction rate after 10 epochs where the standard deviation and mean error are significantly larger (Fig. 4a), these results represent significant improvement and are consistent with previous studies[9]. Furthermore, water extraction rate vs. time plots predicted by the surrogate models agree with the *CMG-GEM* simulations (Fig. 6c).


Train: R^2=0.99844
Test: R^2=0.99858
Validate: R^2=0.99895
MLP
Train: R^2=0.99837
Test: R^2=0.99855
Validate: R^2=0.98478
CNN
Train: R^2=0.99878
Test: R^2=0.99873
Validate: R^2=0.99881
LSTM
Train: R^2=0.99894
Test: R^2=0.99885
Validate: R^2=0.99898
GRU
Predicted water extraction rate (STB/day)
Actual water extraction rate (STB/day)

(a)


Train: σ=3.04e-01, μ=3.59e-03
Test: σ=2.96e-01, μ=3.80e-03
Validate: σ=1.68e-01, μ=9.66e-04
MLP
Train: σ=3.11e-01, μ=1.80e-03
Test: σ=2.99e-01, μ=1.09e-03
Validate: σ=6.41e-01, μ=-8.74e-03
CNN
Train: σ=2.69e-01, μ=1.38e-02
Test: σ=2.80e-01, μ=1.17e-02
Validate: σ=1.79e-01, μ=4.94e-03
LSTM
Train: σ=2.51e-01, μ=-1.43e-02
Test: σ=2.66e-01, μ=-1.22e-02
Validate: σ=1.66e-01, μ=-1.18e-02
GRU
Percent of total grid blocks count
Percent of total grid block count
Prediction error of water extraction rate (STB/day)

(b)



(c)

Fig. 6. Comparisons between the original data and predictions of water extraction rate at the producer by the four surrogate models: (a) grid-block-by-grid-block concordance plots at all simulation times and at production well locations showing R^2 coefficients; (b) grid-block-by-grid-block error distribution showing the standard deviations (σ) and mean errors (μ); (c) sample water extraction rate vs. time plots for each data set. The total number of data points shown in (a) and (b) comprises 2.40×10^6, 3.99×10^5, and 7.99×10^5 for training, validation, and testing, respectively.

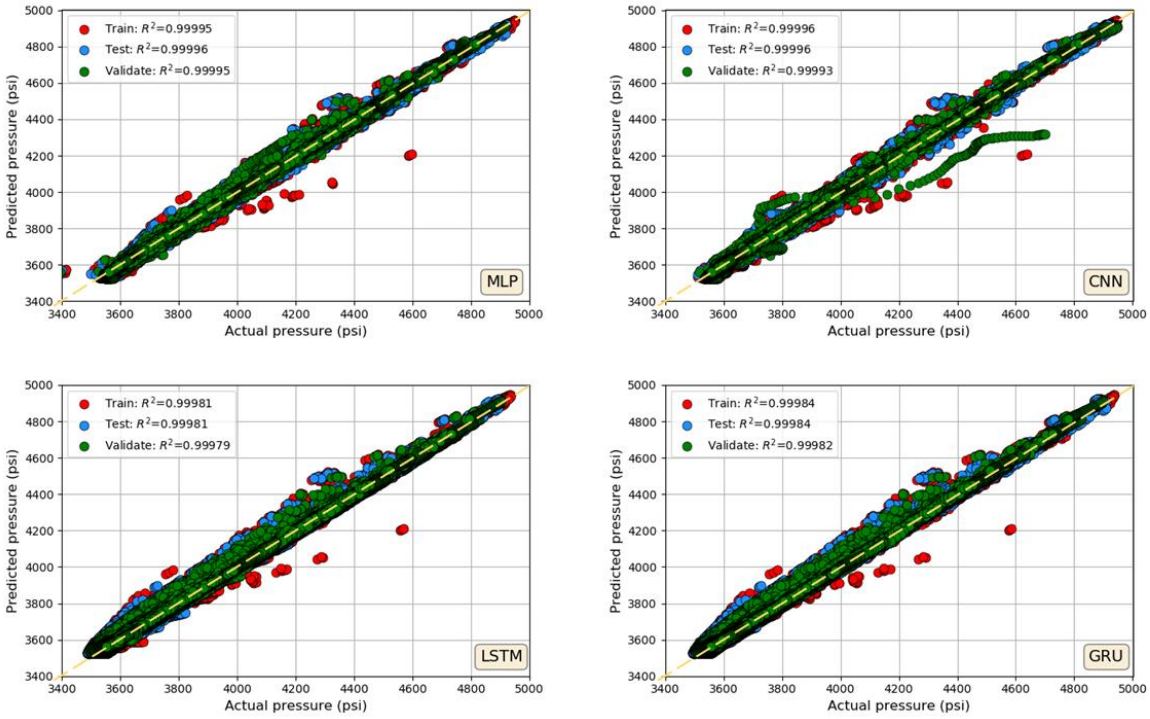
3.3.3 Spatio-Temporal Evolution of Field Pressure and Saturation Plumes

Finally, we investigate the accuracy of predictions of the reservoir state variables (pressure and saturation) by the proxy models both in space and time. Fig. 7 shows comparison of proxy model predictions with traditional full-physics numerical simulation for reservoir pressure at all simulation times on a grid-block-by-grid-block basis. For proper perspective, this corresponds to approximately 2.40×10^6, 3.99×10^5, and 7.99×10^5 data points for training, validation, and testing respectively. The concordance plot for each of the algorithms shows good agreement between the predicted and actual reservoir pressure at various locations and times with near-perfect R^2 coefficients. The corresponding error distributions show additional evidence of outstanding model performance with the standard deviations and mean errors for the four proxy models ranging between 4.64 – 10.70 psi and -1.10 – -1.20 psi, respectively, where average pressure is approximately 4200 psi. Overall, the errors are very small which is very impressive considering the number of data points presented in these plots. Relative to the prediction error state after 10 epochs (Fig. 4a), these final near-zero error distributions demonstrate convergence of the deep-learning solutions.

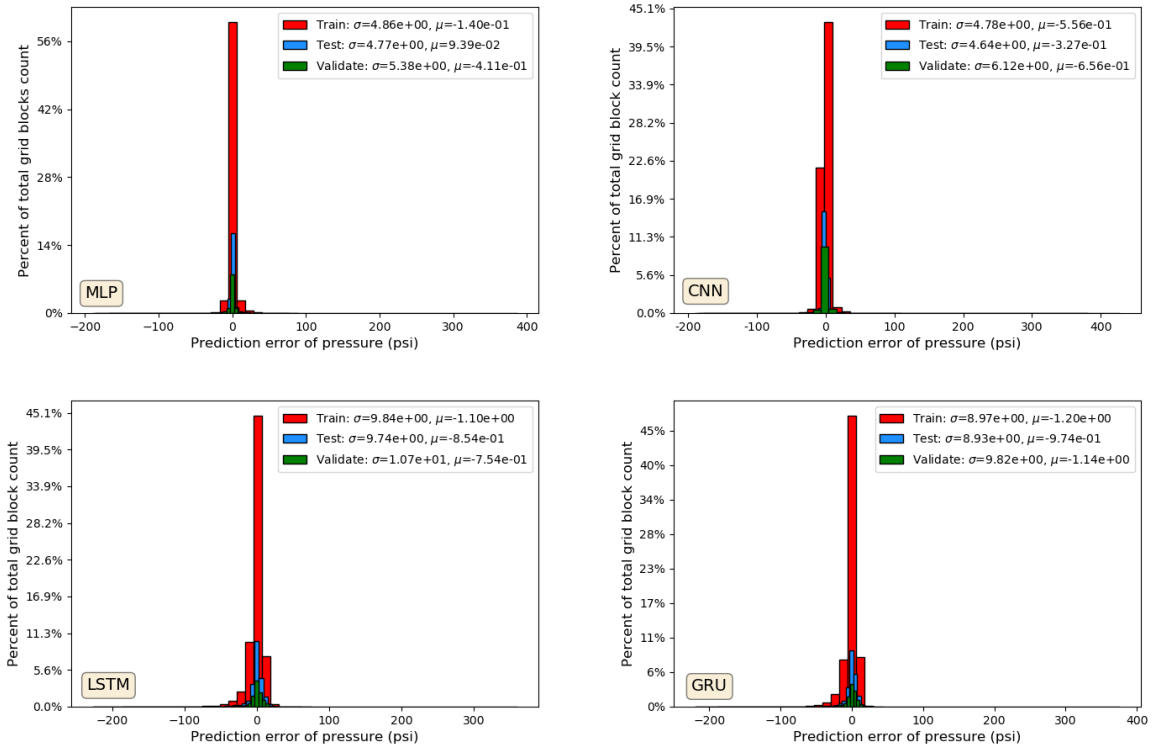
Recent advances in assessment of image quality have shown that structural similarity index (SSIM) is a more accurate measure of perceived similarity between two images than commonly used MSE because SSIM determines whether or not two images are the same based on their texture[31, 32]. SSIM value of 1 means that the images are very similar but a value of 0 means that they are not. As an illustration, we first computed MSE and SSIM on three pairs of pressure contours that are chosen in the following ways: (i) a pair of images comprising two randomly selected normalized *CMG-GEM*-simulated pressure contours that are copies of each other, (ii) a pair of images similar to (i) but with the second image containing some noise, and (iii) a pair of images similar to (ii) but with a constant added to the second image instead of noise. Cases (ii) and (iii) are essentially two modifications to the baseline pressure contour in (i). As shown in Fig. 7c, MSE = 0.00 and SSIM = 1.00 for Case (i) because a pair of duplicated images was used and thus the images are 100% similar based on the SSIM value. For Case (ii), MSE = 0.06 and SSIM = 0.23 due to the noise added to the second image. Finally, MSE = 0.06 and SSIM = 0.87 for Case (iii). Comparison between Cases (ii) and (iii) shows that modifications of the original pressure contour result in two contours that have the same MSE but different SSIMs. Based on MSE values alone, Cases (ii) and (iii) would have been erroneously interpreted as demonstrating the same accuracies relative to the original image in Case (i). The SSIMs for these cases are however lower than SSIM of the perfectly similar contours in Case (i) due to the imposed alterations. Obviously, the SSIM value for Case (iii) agrees with visual inspection which shows that this image is more similar to the baseline image in Case (i) than the image in Case (ii).

Following the above observation, we used SSIM to determine whether the predicted 25 x 25 2D images of the reservoir pressure at each of the three horizontal layers in our grid-based stencil are analogous to the *CMG-GEM* full-physics pressure contours. To do this, we estimated SSIM between corresponding inverted image pairs predicted from each of the four proxy models and the original *CMG-GEM* pressure contours and then generated a cloud plot, which we refer to

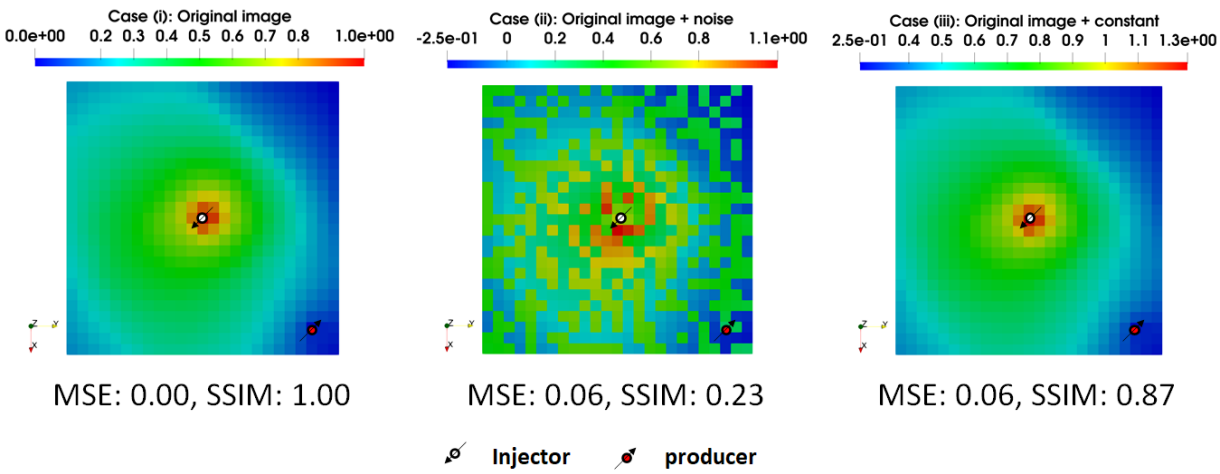
here as the “icicle” plot (Fig. 7d). For proper perspective, each icicle plot shows the SSIM value for a pair of 3834, 639, and 1278 25 x 25 2D images from training, validation, and testing sets, respectively. The number of images in each data set is obtained by multiplying the number of static models in the set with the number of time steps and the number of horizontal layers in our Cartesian grid. To interpret these plots, we set a threshold SSIM value of 0.95 and determine how many SSIM values are equal to or greater than this threshold. We then express this as a percentage of the total count and refer to this as the SSIM image-based accuracy of our proxy models. These accuracies are presented in the plot along with the mean SSIMs. Estimated image-based model accuracies for MLP and CNN proxy models are very close to those presented in Table 2 with the least accurate still above 97%. The corresponding least accurate estimates for the LSTM and GRU proxy models is above 90%. Results from these 2D spatio-temporal analyses support the 1D spatio-temporal outcomes from the concordance plots and error distributions. 3D contours of the reservoir pressure at the final time step for four (out of the six) testing data sets are presented in Fig. 7e along with the prediction error. It is worth noting that these data sets were never-before-seen by the proxy models during the course of training. The range of prediction errors displayed in the color bars further confirms good agreement between the proxy models and full-physics numerical simulations.


Train: R^2=0.99995
Test: R^2=0.99996
Validate: R^2=0.99995
MLP
Train: R^2=0.99996
Test: R^2=0.99996
Validate: R^2=0.99993
CNN
Train: R^2=0.99981
Test: R^2=0.99981
Validate: R^2=0.99979
LSTM
Train: R^2=0.99984
Test: R^2=0.99984
Validate: R^2=0.99982
GRU
Predicted pressure (psi)
Actual pressure (psi)

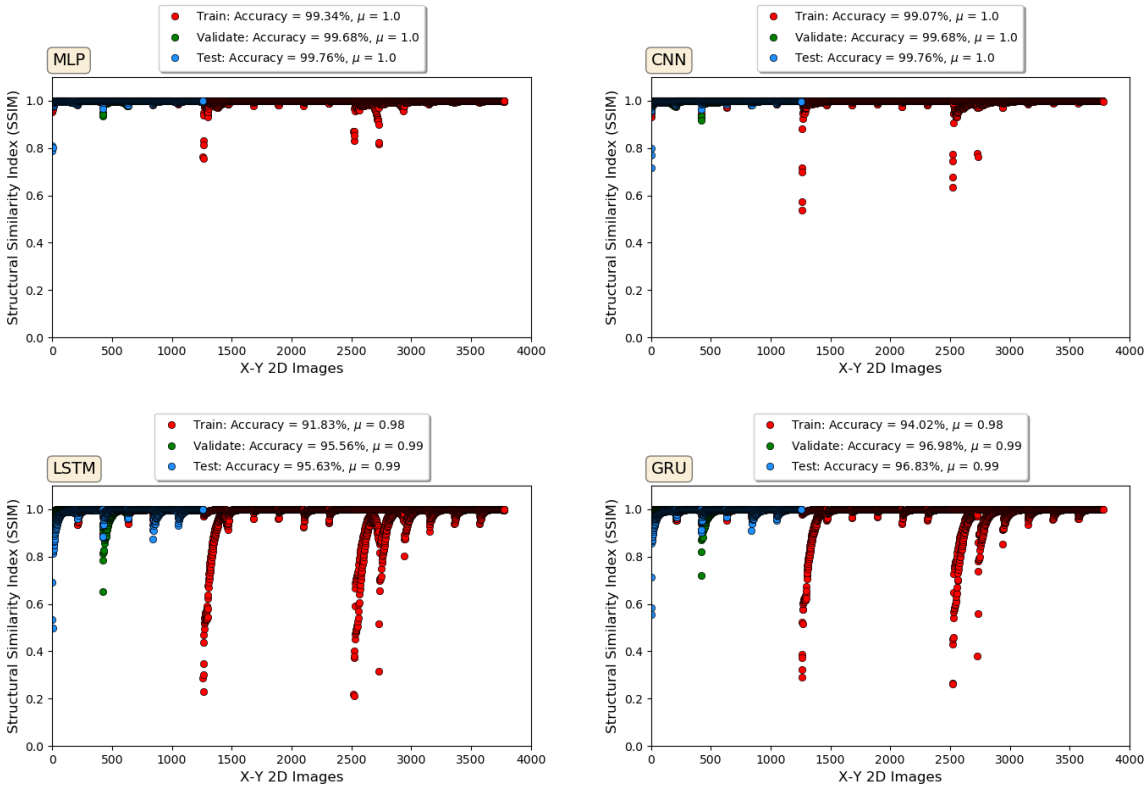
(a)


Train: σ=4.86e+00, μ=-1.40e-01
Test: σ=4.77e+00, μ=9.39e-02
Validate: σ=5.38e+00, μ=-4.11e-01
MLP
Train: σ=4.78e+00, μ=-5.56e-01
Test: σ=4.64e+00, μ=-3.27e-01
Validate: σ=6.12e+00, μ=-6.56e-01
CNN
Train: σ=9.84e+00, μ=-1.10e+00
Test: σ=9.74e+00, μ=-8.54e-01
Validate: σ=1.07e+01, μ=-7.54e-01
LSTM
Train: σ=8.97e+00, μ=-1.20e+00
Test: σ=8.93e+00, μ=-9.74e-01
Validate: σ=9.82e+00, μ=-1.14e+00
GRU
Percent of total grid blocks count
Percent of total grid block count
Prediction error of pressure (psi)

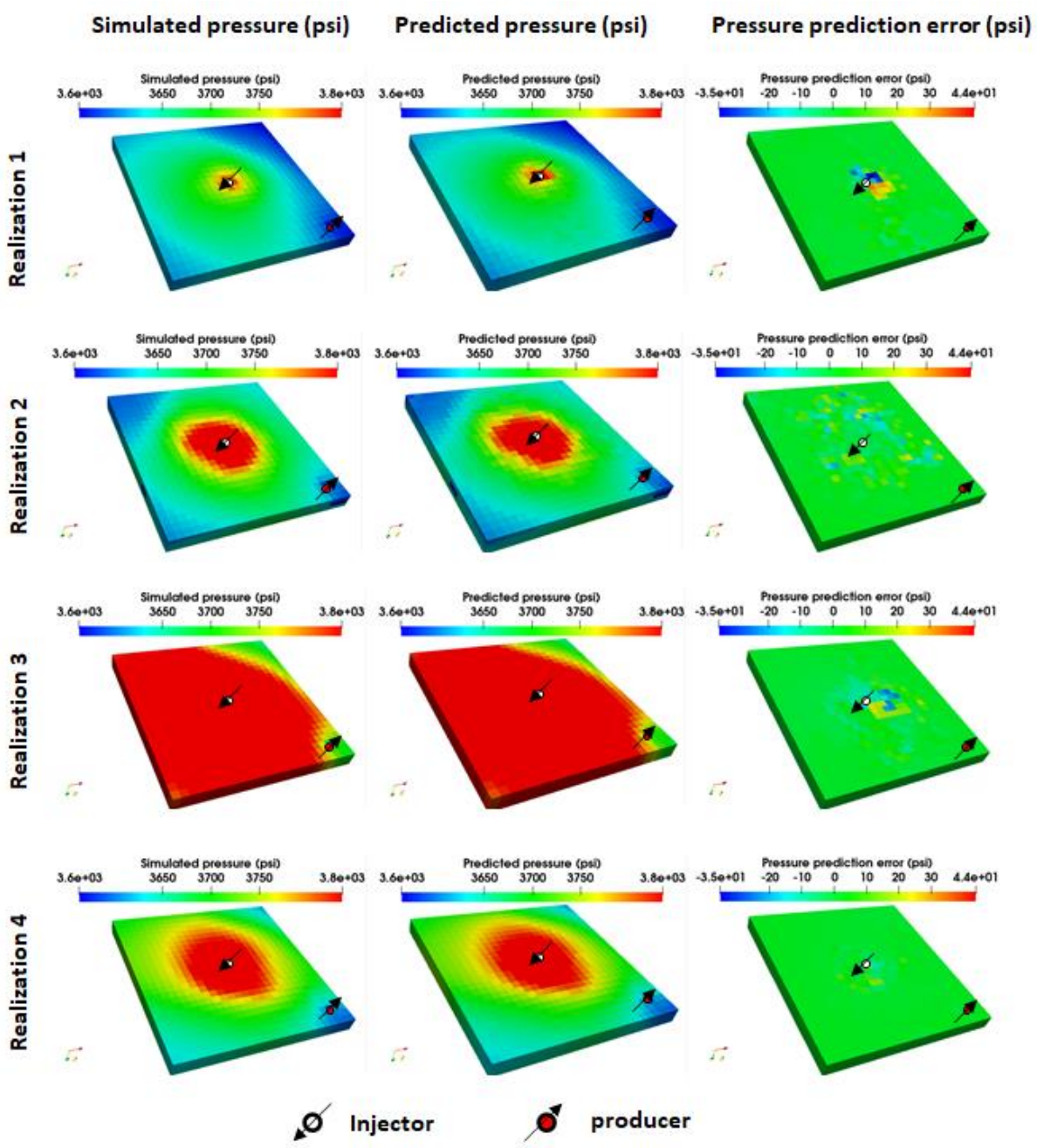
(b)


Case (i): Original image
0.0e+00 0.2 0.3 0.4 0.5 0.6 0.7 0.8 1.0e+00
Case (ii): Original image + noise
-2.5e-01 0 0.2 0.4 0.6 0.8 1.1e+00
Case (iii): Original image + constant
2.5e-01 0.4 0.5 0.6 0.7 0.8 0.9 1 1.1 1.3e+00
MSE: 0.00, SSIM: 1.00
MSE: 0.06, SSIM: 0.23
MSE: 0.06, SSIM: 0.87
Injector
producer

(c)


MLP
Train: Accuracy = 99.34%, μ = 1.0
Validate: Accuracy = 99.68%, μ = 1.0
Test: Accuracy = 99.76%, μ = 1.0
CNN
Train: Accuracy = 99.07%, μ = 1.0
Validate: Accuracy = 99.68%, μ = 1.0
Test: Accuracy = 99.76%, μ = 1.0
LSTM
Train: Accuracy = 91.83%, μ = 0.98
Validate: Accuracy = 95.56%, μ = 0.99
Test: Accuracy = 95.63%, μ = 0.99
GRU
Train: Accuracy = 94.02%, μ = 0.98
Validate: Accuracy = 96.98%, μ = 0.99
Test: Accuracy = 96.83%, μ = 0.99
Structural Similarity Index (SSIM)
X-Y 2D Images

(d)

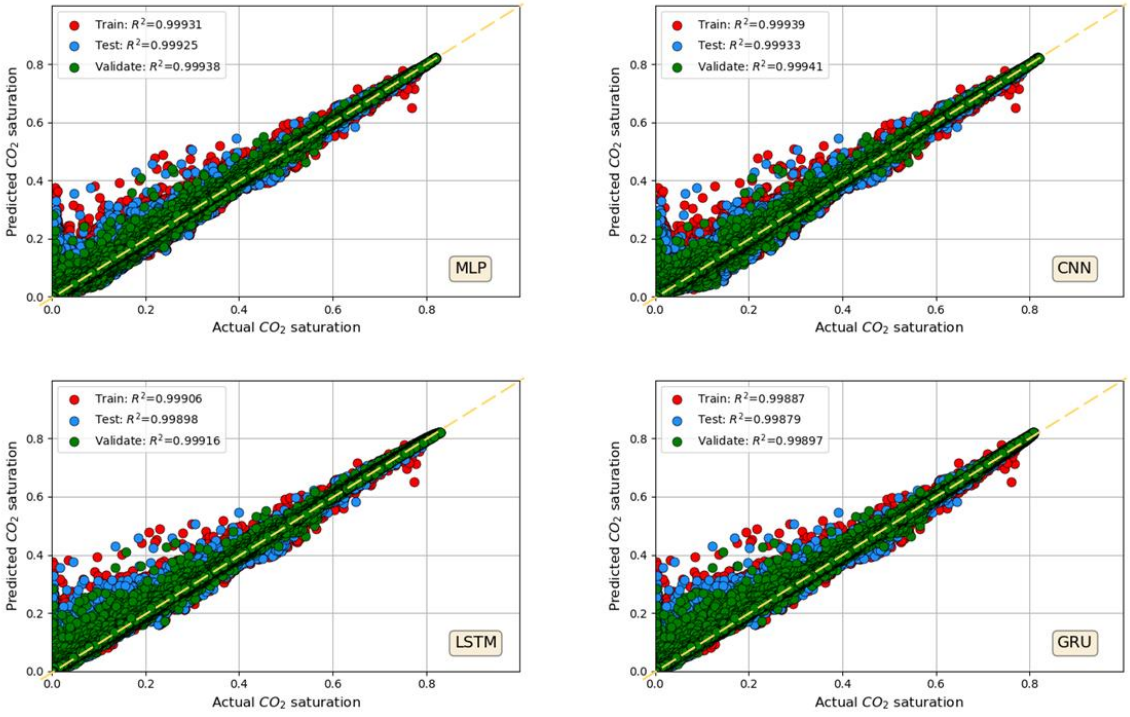


(e)

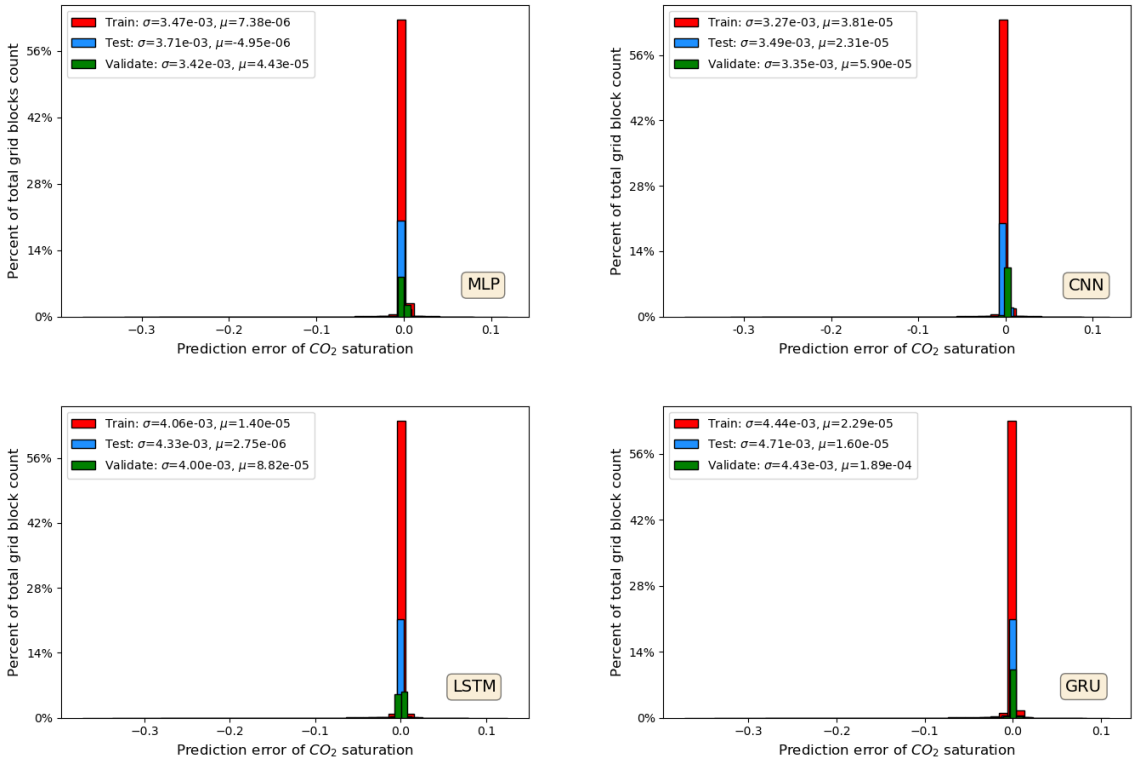
Fig. 7. Comparisons between the original data and predictions of reservoir pressure plume by the four surrogate models: (a) grid-block-by-grid-block concordance plots at all simulation times and locations showing R^2 coefficients; (b) grid-block-by-grid-block error distribution showing the standard deviations (σ) and mean errors (μ); (c) randomly-selected pressure contour showing comparison between MSE and SSIM; (d) "icicle" plots showing SSIM computed from the predicted and the original 2D images of pressure contours; (e) contours of reservoir pressure at the final time step for four static model

realizations from the testing data sets showing (from left to right) full-physics simulations, neural network predictions, and the prediction errors. The total number of data points shown in (a) and (b) comprises 2.40×10^6, 3.99×10^5, and 7.99×10^5 for training, validation, and testing, respectively.

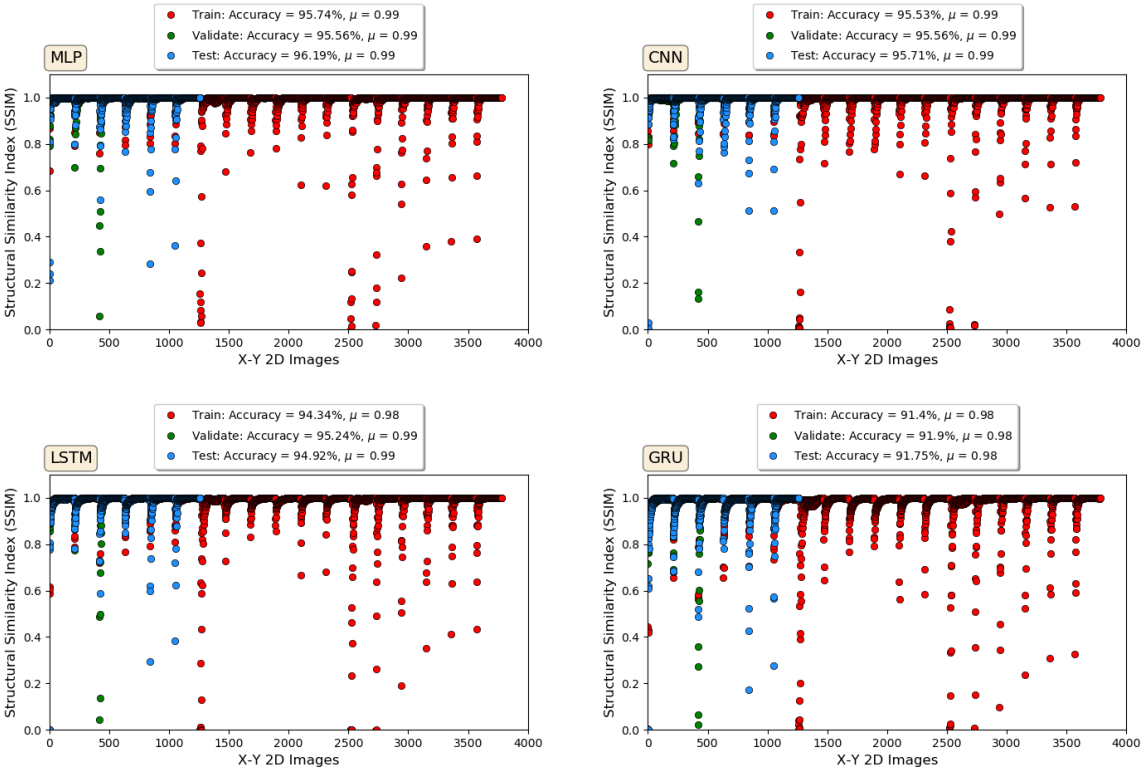
Similar results were obtained for CO_2 saturation as shown in Fig. 8. Specifically, predicted CO_2 saturations are in good agreement with *CMG-GEM* simulations on a grid-block-by-grid-block basis as supported by the R^2 values. The range of σ and μ values in the error distribution provides additional evidence for the satisfactory performance of the models. SSIM imaged-based model accuracies that were estimated with 2D images of CO_2 saturation are also reasonable. Finally, 3D contours of CO_2 saturation plume using never-before-seen data sets show practically negligible prediction errors between the proxy models and the *CMG-GEM* simulations, with the highest errors occurring at the CO_2-water interface. Generally, the surrogate models satisfactorily predict spatio-temporal evolution of reservoir pressure and CO_2 saturation surfaces across the entire simulation domain.



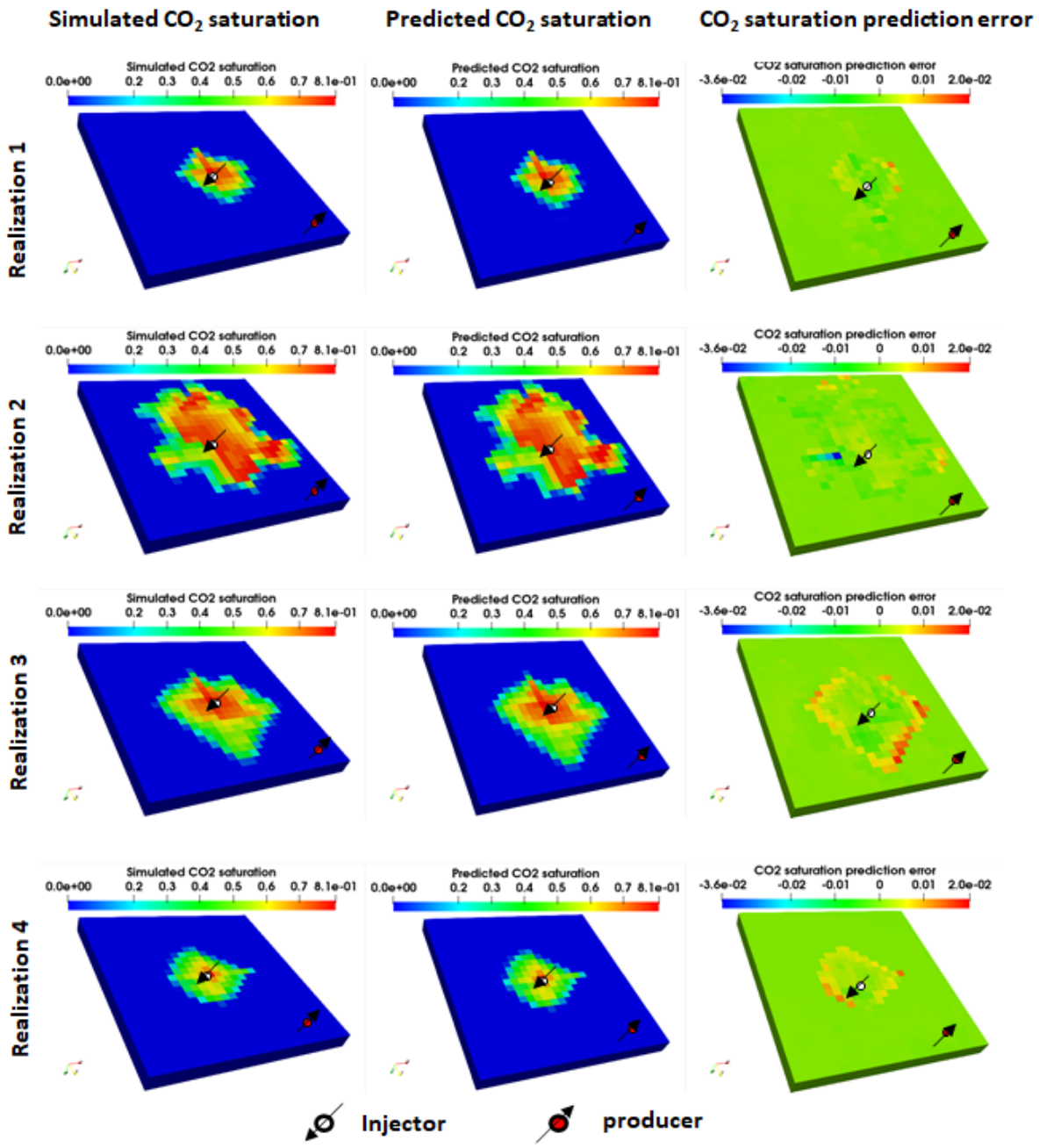
(a)


Train: σ=3.47e-03, μ=7.38e-06
Test: σ=3.71e-03, μ=-4.95e-06
Validate: σ=3.42e-03, μ=4.43e-05
MLP
Train: σ=3.27e-03, μ=3.81e-05
Test: σ=3.49e-03, μ=2.31e-05
Validate: σ=3.35e-03, μ=5.90e-05
CNN
Train: σ=4.06e-03, μ=1.40e-05
Test: σ=4.33e-03, μ=2.75e-06
Validate: σ=4.00e-03, μ=8.82e-05
LSTM
Train: σ=4.44e-03, μ=2.29e-05
Test: σ=4.71e-03, μ=1.60e-05
Validate: σ=4.43e-03, μ=1.89e-04
GRU
Percent of total grid blocks count
Percent of total grid block count
Prediction error of CO_2 saturation

(b)


Train: Accuracy = 95.74%, μ = 0.99
Validate: Accuracy = 95.56%, μ = 0.99
Test: Accuracy = 96.19%, μ = 0.99
MLP
Train: Accuracy = 95.53%, μ = 0.99
Validate: Accuracy = 95.56%, μ = 0.99
Test: Accuracy = 95.71%, μ = 0.99
CNN
Train: Accuracy = 94.34%, μ = 0.98
Validate: Accuracy = 95.24%, μ = 0.99
Test: Accuracy = 94.92%, μ = 0.99
LSTM
Train: Accuracy = 91.4%, μ = 0.98
Validate: Accuracy = 91.9%, μ = 0.98
Test: Accuracy = 91.75%, μ = 0.98
GRU
Structural Similarity Index (SSIM)
X-Y 2D Images

(c)



(d)

Fig. 8. Comparisons between the original data and predictions of CO_2 saturation plume by the four surrogate models: (a) grid-block-by-grid-block concordance plots at all simulation times and locations showing R^2 coefficients; (b) grid-block-by-grid-block error distribution showing the standard deviations (σ) and mean errors (μ); (c) "icicle" plots showing SSIM computed from the predicted and the original 2D images of CO_2 saturation contours; (d) contours of CO_2 saturation at the final time step for four static model realizations from the testing data sets showing (from left to right) full-physics simulations, neural

network predictions, and the prediction errors. The total number of data points shown in (a) and (b) comprises 2.40×10^6, 3.99×10^5, and 7.99×10^5 for training, validation, and testing, respectively.

3.3.4 Forecasting reservoir pressure, CO_2 saturation, water extraction rate, and BHP at the CO_2 well

Technically, model prediction is analogous to model forecasting. Forecasting in our context implies the prediction of pressure, CO_2 saturation, water extraction rate, and bottomhole pressure at the CO_2 injection well for simulation times beyond the time limit of the sets of data used for model training. So, we evaluated model forecasting using the MLP model that was trained with only the first 36 months of training data, followed by prediction of 72 months of data that comprises both the first and last 36 months of the blind test datasets.

Comparison of the MLP model predictions with CMG simulations (actual) is presented in Fig. 9. Concordance plots of predicted vs. actual reservoir pressure, CO_2 saturation, and water extraction rate show good match with very high R^2 scores. Standard deviations and mean errors of the error distributions for the blind test cases of each predicted variable are 6.54 psi, 4.66×10^{-3}, and 0.273 STB/day and 3.04 psi, 8.50×10^{-4}, and -6.49×10^{-4} STB/day, respectively. These values, especially for those of reservoir pressure and CO_2 saturation, are slightly higher than the corresponding standard deviations of predictions made with the MLP model that was trained with the complete 72 months of training data (please refer to Figs. 6b, 7b, and 8b). However, these errors are negligible considering the number of data points presented and the range of each data. Plots of water extraction rate at the producer and bottomhole pressure at the injector vs. time show excellent agreement between the MLP model forecasts and the CMG simulations (Fig. 9c). Therefore, the proposed methodology is very efficient for forecasting the future state of the reservoir and well rates and bottomhole pressure.

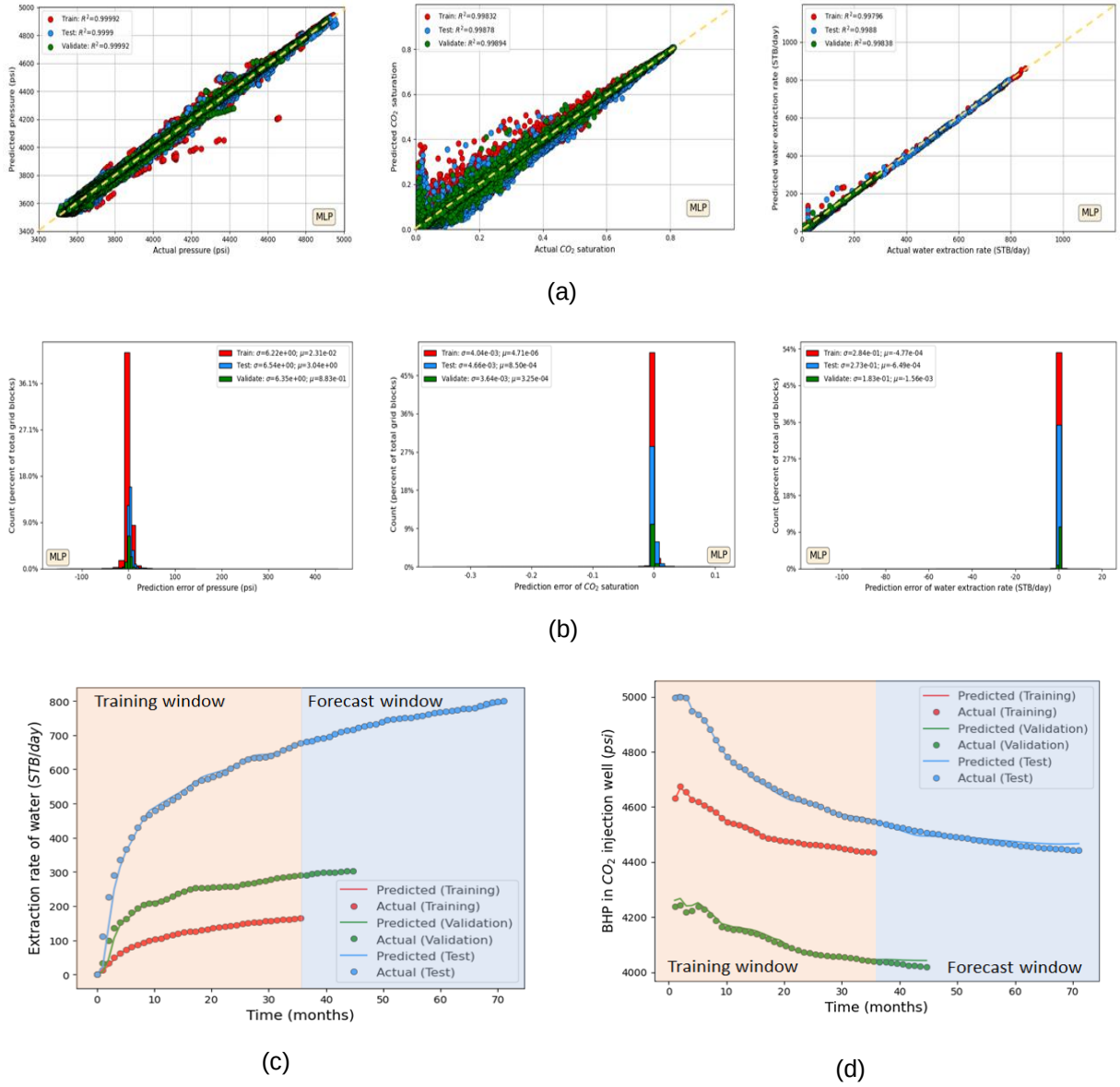


Fig. 9. MLP model predictions of reservoir pressure, CO_2 saturation, and water extraction rate using sub-sampled simulation times for model training and validation (36, 45, and 72 months of data were used for training, validation, and blind testing respectively): (a) grid-block-by-grid-block concordance plots at all simulation times and at production well locations showing R^2 coefficients for reservoir pressure, CO_2 saturation, and water extraction rate; (b) grid-block-by-grid-block error distribution showing the standard deviations (σ) and mean errors (μ) for reservoir pressure, CO_2 saturation, and water extraction rate; (c) sample water extraction rate at the producer vs. time plots for each data subset showing forecasting of rate (using the blind test data) beyond the 36 months of training data; (d) sample bottomhole pressure at the CO_2 injection well vs. time plots for each data subset showing forecasting of bottomhole pressure (using the blind test data) beyond the 36 months of training data. The total number of data points shown

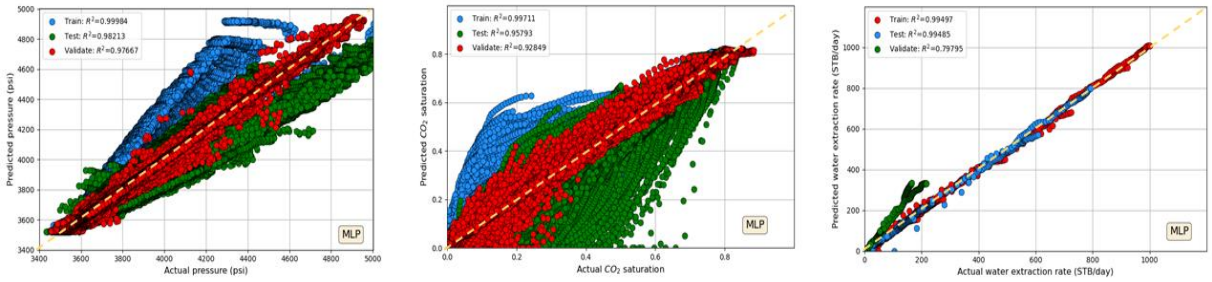
in (a) and (b) comprises $1.22x10^6$, $2.53x10^5$, and $7.99x10^5$ for training, validation, and testing, respectively.

3.4 Performance Comparison between Physics-Framed and Traditional Deep-Learning Approaches

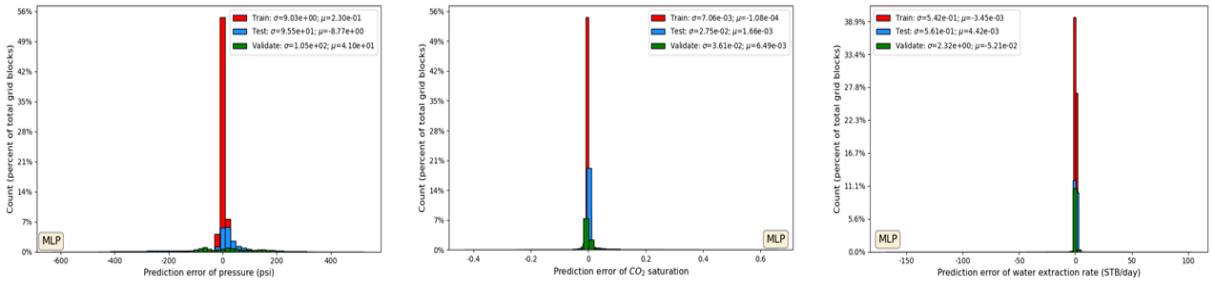
The traditional deep-learning approach is not constrained or guided by physical laws during model development. To evaluate the performance of the proposed methodology compared to a traditional deep-learning approach, a five-layer MLP model was trained without the use of historical data as inputs. The only input to the model is the static data that comprises the rock properties (i.e., porosity and directional permeabilities), grid-block sizes, well constraints (i.e., constant injection rate and BHP), and simulation times. The model outputs are reservoir pressure, CO_2 saturation, and water extraction rate.

Fig. 10 shows the performance of the five-layer MLP model. While the model seems to reasonably predict water extraction rate with the training and blank test datasets (please refer to the third plot in Fig. 10a), albeit to a lesser degree with the validation dataset, it falls short in properly matching the spatio-temporal evolution of reservoir pressure and CO_2 saturation as a significant number of the data points evidently do not correctly align with the dashed 1:1 (i.e., 45°) line and the R^2 scores are slightly lower when compared to Figs. 6a, 7a, and 8a. In addition, note that the training data (i.e., the red data points in the plots) line up better with the 1:1 line than the validation and blank test data both of which were not used by the MLP algorithm during forward and backward propagation but often are the litmus test for how well the model generalizes to all the three sets of data. This is a clear evidence that the model was overfitted, a situation that occurs when an ML model captures patterns in the training data that do not generalize to the complete datasets, thereby performing less well with the blank test data than with the training data. Furthermore, statistics of the error distributions (Fig. 10b), especially the standard deviation, indicate that the data are very dispersed relative to the mean.

Due to the unique nature of subsurface flow data where a structure exists, as dictated by the governing physical law(s), that correlates the discrete variables in both space and time, it is obvious that formulating subsurface flow deep-learning problems using the governing physical law(s) helps to guide model training and development towards a more accurate ML-based surrogate model. Results from the methodology proposed in this study (sections 3.1 through 3.3) honor these requirements, thereby avoiding the limitations with traditional deep-learning approaches.



(a)



(b)

Fig. 10. Predictions of reservoir pressure, CO_2 saturation, and water extraction rate using a five-layer MLP model trained with traditional deep-learning approach: (a) grid-block-by-grid-block concordance plots at all simulation times and at production well locations showing R^2 coefficients for reservoir pressure, CO_2 saturation, and water extraction rate; (b) grid-block-by-grid-block error distribution showing the standard deviations (σ) and mean errors (μ) for reservoir pressure, CO_2 saturation, and water extraction rate. The total number of data points shown in (a) and (b) comprises 2.43×10^6, 4.05×10^5, and 8.10×10^5 for training, validation, and testing, respectively.

4. MODEL EXTENSION

Numerical models are often modified through mesh refinement around wellbores or alteration to the initial and/or boundary conditions; thus, model transferability could become an issue. In the case that meshes are refined, changes in discretization should alter the number or pattern of grid points and also alter the local reservoir state with respect to pore pressure and saturation, whereas production/injection rates will not change significantly. In situations where new grid points are added through mesh refinement, the fact that we carefully sampled our features space, specifically permeability and porosity, and that these formation properties are heterogeneously distributed in space, gives credence to the likelihood that the models will perform satisfactorily at inter-block locations in the original reservoir geometry where new grid points may be added after mesh refinement. This is so because predictions at such location is an interpolation, and not an extrapolation problem for the models. Thus, we expect that our approach will work irrespective of refined meshes near the wells because the dependence of the ML targets for our models (i.e., pore pressure, CO_2 saturation, etc.) on the ML features used in this study is a theoretically valid proposition. In the event that the initial and/or boundary conditions are modified, which would likely alter the pore pressure field, CO_2 saturation plume, and well flow rates, it is not necessary to retrain the entire model from scratch. Rather, transfer learning can easily be used to update the weights and biases of features that were already learned by our pre-trained models without significant loss of generalization. Finally, it is worth noting that our methodology is not incompatible with 3D image-based datasets. Imaged-based training is only a matter of data formatting and has no implications on the importance of applying physical laws to carefully formulate ML-based subsurface flow problems. Because of the modest size of the reservoir geometry used to demonstrate our approach, we directly trained the model at the grid blocks without first extracting key features which is typically the case with datasets that are prepared in 3D image formats. This has the added advantage of minimizing the network capacity and thus, the training time. Unlike other approaches where a PDE loss is added to the neural network

objective function and less than 2% of the reservoir grid blocks are randomly sampled and used for model training due to the high computational burden of computing the PDE loss on the full reservoir geometry (e.g., Raissi et al.[14, 15]), our approach uses all the grid blocks in order to properly capture the spatial heterogeneity in the data; thus, can readily be scaled to large-scale reservoir geometries.

5. CONCLUSIONS

Deep-learning-based proxy models have been developed for CCUS using off-the-shelf supervised machine-learning algorithms applied to a physics-guided subsurface two-phase flow problem involving immiscible displacement of water by CO_2. The models reasonably satisfy the underlying physical laws governing the transport of the wetting and nonwetting fluids and are therefore excellent approximations of the full-physics analogue. In addition, these surrogate models are lean and very robust, simultaneously predicting reservoir pressure and CO_2 saturation, including the surface well flow rate and bottomhole pressure, at all simulation times in just a few seconds on a standard desktop computer. Model testing with never-before-seen data shows satisfactory performance, with the MLP model outperforming the other models in training speed and prediction time. A key outcome of this study is that limits can be placed on network design parameters to avoid over designing neural networks, with associated efficiencies in training and prediction times.

The models developed in this study can be used as fast forward models during history-matching, replacing computationally expensive full-physics models and thereby allowing near real-time forecasts of subsurface processes that are critical for supporting rapid decision making throughout the life cycle of CCUS operations. Furthermore, large-scale deployment of CCUS technology often requires calibration of numerical models with field data to build a representative reservoir/hydrogeologic model of the subsurface, which is critical for project management and interpretation of long-term monitoring data, during the post-injection period. Recognizing that DL-

based proxy models trained with pure simulation data may not necessarily capture the response from real reservoirs, which is also a potential pitfall of full-physics numerical models, these proxy models are retrainable and can be updated using ongoing field observations through transfer learning. The retrained models then become surrogates for actual reservoirs that could be used to rapidly query the critical flow dynamics needed to quickly address stakeholders' concerns on issues such as the potential for induced seismicity, vertical fluid migration through caprocks, CO_2 leakage into freshwater aquifers from compromised wells, etc. In addition, these surrogate models can be incorporated in a virtual learning environment to be used by operators for optimizing reservoir development prior to field activities, by regulators during processing of permits, and by the public to gain intuitive understanding and rapid insight into subsurface carbon storage. The improved ability to predict and understand subsurface CCUS processes and impacts in general provides the foundation for overall better decision-making including faster permitting that will help lower current barriers to commercial-scale deployment of CCUS technology and ultimately bolster public confidence in CCUS projects.

Acknowledgements

This work was completed as part of the Science-informed Machine learning to Accelerate Real Time decision making for Carbon Storage (SMART-CS) Initiative (edx.netl.doe.gov/SMART). Support for this initiative was provided by the U.S. Department of Energy's (DOE) Office of Fossil Energy's Carbon Storage Research program through the National Energy Technology Laboratory (NETL). The authors wish to acknowledge Mark McKoy (NETL, Carbon Storage Technology Manager), Darin Damiani (DOE Office of Fossil Energy, Carbon Storage Program Manager), and Mark Ackiewicz (DOE Office of Fossil Energy, Director, Division of Carbon Capture and Storage Research and Development), for programmatic guidance, direction and support. The authors also wish to acknowledge the SMART-CS advisory board for their support and contributions, Seyyed Hosseini of the Bureau of Economic Geology, Jackson School of Geosciences, University of

Texas at Austin, for providing the data used in this study, and Diana Bacon of the Pacific Northwest National Laboratory for providing python scripts used to process CMG-GEM output files into numpy arrays.

Appendix: Guiding the formulation of subsurface deep-learning problems with fluid transport equations

To formulate a problem statement for the machine-learning algorithms used in this study, we begin by examining a simple case of the partial differential equations governing fluid flow through porous media.

Appendix A: Single-phase fluid flow

Flow of single-phase fluids (e.g., water) through porous media is governed by the diffusivity equation below:

$$\phi c_r \frac{\partial p}{\partial t} + \nabla \bullet \left(\bar{v} \right) = \bar{q} \quad \text{..... (A1),}$$

where,

$$\bar{v} = -\frac{\overline{\overline{\mathbf{K}}}}{\mu} \nabla p; \quad \overline{\overline{\mathbf{K}}} = \begin{bmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & k_z \end{bmatrix}$$

Equation A1 assumes constant fluid densities and viscosities, zero capillary pressure, no gravity effect, isothermal conditions, and uniform grids. Using two-point flux approximation discretization scheme, the implicit finite-difference formulation for Equation A1 is:

$$p_{i,j,k}^{n+1} - \frac{\Delta t}{\phi c_r} \left(\frac{k_x}{\mu} \frac{p_{i+1,j,k}^{n+1} - 2p_{i,j,k}^{n+1} + p_{i-1,j,k}^{n+1}}{\Delta x^2} + \frac{k_y}{\mu} \frac{p_{i,j+1,k}^{n+1} - 2p_{i,j,k}^{n+1} + p_{i,j-1,k}^{n+1}}{\Delta y^2} + \frac{k_z}{\mu} \frac{p_{i,j,k+1}^{n+1} - 2p_{i,j,k}^{n+1} + p_{i,j,k-1}^{n+1}}{\Delta z^2} + \bar{q} \right) = p_{i,j,k}^{n}$$

..... (A2)

799 $$\alpha p_{i,j,k}^{n+1} + \beta^{+1} p_{i+1,j,k}^{n+1} + \beta^{-1} p_{i-1,j,k}^{n+1} + \gamma^{+1} p_{i,j+1,k}^{n+1} + \gamma^{-1} p_{i,j-1,k}^{n+1} + \omega^{+1} p_{i,j,k+1}^{n+1} + \omega^{-1} p_{i,j,k-1}^{n+1} = p_{i,j,k}^{n} \quad \text{..... (A3),}$$

800 where,

801 $$\alpha = 1 + \frac{\Delta t}{\phi \mu c_r}\left(\frac{2k_x}{\Delta x^2} + \frac{2k_y}{\Delta y^2} + \frac{2k_z}{\Delta z^2}\right);\ \beta^{+1} = -\frac{\Delta t}{\phi \mu c_r}\frac{k_x}{\Delta x^2};\ \beta^{-1} = -\frac{\Delta t}{\phi \mu c_r}\frac{k_x}{\Delta x^2};\ \gamma^{+1} = -\frac{\Delta t}{\phi \mu c_r}\frac{k_y}{\Delta y^2};\ \gamma^{-1} = -\frac{\Delta t}{\phi \mu c_r}\frac{k_y}{\Delta y^2};$$
$$\omega^{+1} = -\frac{\Delta t}{\phi \mu c_r}\frac{k_z}{\Delta z^2};\ \omega^{-1} = -\frac{\Delta t}{\phi \mu c_r}\frac{k_z}{\Delta z^2}$$

802 When assembled into a linear system of equations, the algebraic equation in A3 results into:

803 $$[\mathbf{A}]\{p_I^{n+1}\} = \{p_I^n\} \quad \text{..... (A4)}$$

804 Future time-step pressure solutions in every grid block is obtained from prior time steps as
805 follows:

806 $$\{p_I^{n+1}\} = [\mathbf{A}]^{-1}\{p_I^n\} \quad \text{..... (A5)}$$

807 $[\mathbf{A}]$ is a sparse matrix whose sparsity pattern depends on the dimension of the problem (i.e., 1D,
808 2D, or 3D), grid type, boundary conditions (inner and outer), etc. Evidently from Equation A3,
809 values of matrix elements in $[\mathbf{A}]$ depend on time-step size (Δt), grid-block size (Δx, Δy, Δz),
810 rock properties (rock compressibility, c_r; porosity, ϕ; directional permeabilities, k_x, k_y, k_z), and
811 fluid viscosity, μ. Therefore, Equation A5 shows that future time-step pressure solutions depend
812 on pressure solutions at the prior time steps, time-step size, grid-block size, well and outer
813 boundary constraints, and rock and fluid properties. With $\{p_I^{n+1}\}$ designated as output and $\{p_I^n\}$,
814 as well as the variables needed to assemble the sparse matrix $[\mathbf{A}]$ (i.e.,
815 Δx, Δy, Δz, Δt, ϕ, k_x, k_y, k_z, c_r, and μ), designated as inputs, we can demarcate features and
816 targets for artificial neural network algorithms. The algorithms are essentially tasked to learn the

best representation of sparse matrix [*A*] that honors Equation A3 at all the time steps in the data sets.

Appendix B: Two-phase fluid flow

The continuity equation for multiphase flow of fluids (e.g., CO_2 and water) through porous media is described as:

$$\frac{\partial}{\partial t}\left(\phi\rho_\alpha S_\alpha\right)+\nabla\bullet\left(-\frac{\rho_\alpha\overline{\overline{\mathbf{K}}}k_{r\alpha}}{\mu_\alpha}\nabla\left(p_\alpha-\rho_\alpha gz\right)\right)=\rho_\alpha\bar{q}_\alpha;\quad \alpha=\mathrm{CO}_2,\ \text{water} \quad \text{(B1)}$$

In this study, constant fluid densities and viscosities, zero capillary pressure, no gravity effect, isothermal conditions, and uniform grids in the x- and y-directions are assumed. Adding the wetting and non-wetting phase continuity equations results in the following pressure and transport equations.

$$\phi c_r\frac{\partial p}{\partial t}+\nabla\bullet\left(\bar{v}\right)=\bar{q}_t \quad \text{(B2)},$$

$$\phi\frac{\partial S_w}{\partial t}+\phi c_r S_w\frac{\partial p}{\partial t}+\nabla\bullet\left(f_w\bar{v}\right)=\bar{q}_w;\quad w=\text{water} \quad \text{(B3)},$$

where,

$$\bar{q}_t=\bar{q}_w+\bar{q}_{nw}$$

$$\lambda_t=\lambda_w+\lambda_{nw}=\frac{k_{rw}}{\mu_w}+\frac{k_{rnw}}{\mu_{nw}}$$

$$\bar{v}=\bar{v}_w+\bar{v}_{nw}=-\left(\frac{\overline{\overline{\mathbf{K}}}k_{rw}}{\mu_w}+\frac{\overline{\overline{\mathbf{K}}}k_{rnw}}{\mu_{nw}}\right)\nabla p=-\overline{\overline{\mathbf{K}}}\lambda_t\nabla p$$

$$f_w=\frac{\lambda_w}{\lambda_t}$$

IMPES (Implicit pressure explicit saturation) discrete formulation for the pressure equation is:

835 $$p_{i,j,k}^{n+1} - \frac{\Delta t}{\phi c_r}\left[\begin{array}{l} k_x \dfrac{\lambda_{t_{i+\frac{1}{2},j,k}}^{n} p_{i+1,j,k}^{n+1} - \left(\lambda_{t_{i+\frac{1}{2},j,k}}^{n} + \lambda_{t_{i-\frac{1}{2},j,k}}^{n}\right) p_{i,j,k}^{n+1} + \lambda_{t_{i-\frac{1}{2},j,k}}^{n} p_{i-1,j,k}^{n+1}}{\Delta x^2} + \\ k_y \dfrac{\lambda_{t_{i,j+\frac{1}{2},k}}^{n} p_{i,j+1,k}^{n+1} - \left(\lambda_{t_{i,j+\frac{1}{2},k}}^{n} + \lambda_{t_{i,j-\frac{1}{2},k}}^{n}\right) p_{i,j,k}^{n+1} + \lambda_{t_{i,j-\frac{1}{2},k}}^{n} p_{i,j-1,k}^{n+1}}{\Delta y^2} + \\ k_z \dfrac{\lambda_{t_{i,j,k+\frac{1}{2}}}^{n} p_{i,j,k+1}^{n+1} - \left(\lambda_{t_{i,j,k+\frac{1}{2}}}^{n} + \lambda_{t_{i,j,k-\frac{1}{2}}}^{n}\right) p_{i,j,k}^{n+1} + \lambda_{t_{i,j,k-\frac{1}{2}}}^{n} p_{i,j,k-1}^{n+1}}{\Delta z^2} + \bar{q}_t \end{array}\right] = p_{i,j,k}^{n} \quad \text{..... (B4)}$$

836 $$\alpha p_{i,j,k}^{n+1} + \beta^{+1} p_{i+1,j,k}^{n+1} + \beta^{-1} p_{i-1,j,k}^{n+1} + \gamma^{+1} p_{i,j+1,k}^{n+1} + \gamma^{-1} p_{i,j-1,k}^{n+1} + \omega^{+1} p_{i,j,k+1}^{n+1} + \omega^{-1} p_{i,j,k-1}^{n+1} = p_{i,j,k}^{n} \quad \text{..... (B5),}$$

837 where,

$$\alpha = 1 + \frac{\Delta t}{\phi c_r}\left[\frac{k_x}{\Delta x^2}\left(\lambda_{t_{i+\frac{1}{2},j,k}}^{n} + \lambda_{t_{i-\frac{1}{2},j,k}}^{n}\right) + \frac{k_y}{\Delta y^2}\left(\lambda_{t_{i,j+\frac{1}{2},k}}^{n} + \lambda_{t_{i,j-\frac{1}{2},k}}^{n}\right) + \frac{k_z}{\Delta z^2}\left(\lambda_{t_{i,j,k+\frac{1}{2}}}^{n} + \lambda_{t_{i,j,k-\frac{1}{2}}}^{n}\right)\right];$$

838 $$\beta^{+1} = -\frac{\Delta t}{\phi c_r}\frac{k_x}{\Delta x^2}\lambda_{t_{i+\frac{1}{2},j,k}}^{n};\ \beta^{-1} = -\frac{\Delta t}{\phi c_r}\frac{k_x}{\Delta x^2}\lambda_{t_{i-\frac{1}{2},j,k}}^{n};\ \gamma^{+1} = -\frac{\Delta t}{\phi c_r}\frac{k_y}{\Delta y^2}\lambda_{t_{i,j+\frac{1}{2},k}}^{n};\ \gamma^{-1} = -\frac{\Delta t}{\phi c_r}\frac{k_y}{\Delta y^2}\lambda_{t_{i,j-\frac{1}{2},k}}^{n};$$
$$\omega^{+1} = -\frac{\Delta t}{\phi c_r}\frac{k_z}{\Delta z^2}\lambda_{t_{i,j,k+\frac{1}{2}}}^{n};\ \omega^{-1} = -\frac{\Delta t}{\phi c_r}\frac{k_z}{\Delta z^2}\lambda_{t_{i,j,k-\frac{1}{2}}}^{n}$$

839 Equation B5 can be assembled into the same form of the linear system of equations in Equation
840 A4, where $[\mathbf{A}]$ is a sparse matrix with entries that depend on the wetting fluid saturation inside
841 the grid blocks at the prior time step, time-step size, grid-block size, and rock and fluid properties.
842 Grid-block pressure solutions at the next time step can be obtained using Equation A5. Thus, in
843 addition to grid-block pressure solutions at the prior time steps, time-step size, grid-block size,
844 wellbore and outer boundary constraints, and rock and fluid properties, future time-step pressure
845 solutions in the grid blocks depend also on the grid-block saturation of the wetting fluid at the prior
846 time steps.

847 IMPES formulation for the transport (saturation) equation is:

$$S_{w_{i,j,k}}^{n+1} = \frac{\bar{q}_w + \phi \frac{S_{w_{i,j,k}}^{n}}{\Delta t} - \frac{f_{w_{i+\frac{1}{2},j,k}}^{n} \bar{v}_{i+\frac{1}{2},j,k}^{n+1} - f_{w_{i-\frac{1}{2},j,k}}^{n} \bar{v}_{i-\frac{1}{2},j,k}^{n+1}}{\Delta x} - \frac{f_{w_{i,j+\frac{1}{2},k}}^{n} \bar{v}_{i,j+\frac{1}{2},k}^{n+1} - f_{w_{i,j-\frac{1}{2},k}}^{n} \bar{v}_{i,j-\frac{1}{2},k}^{n+1}}{\Delta y} - \frac{f_{w_{i,j,k+\frac{1}{2}}}^{n} \bar{v}_{i,j,k+\frac{1}{2}}^{n+1} - f_{w_{i,j,k-\frac{1}{2}}}^{n} \bar{v}_{i,j,k-\frac{1}{2}}^{n+1}}{\Delta z}}{\frac{\phi}{\Delta t} + \phi c_r \frac{p_{i,j,k}^{n+1} - p_{i,j,k}^{n}}{\Delta t}}$$

…… (B6),

where, consistent with the two-point flux approximation scheme, the fractional flow (f_w) and Darcy flux ($\bar{v}$) are evaluated at the grid-block boundaries. Consequently, future time step wetting-fluid saturations for every grid block depend on the grid-block fluid saturations at the prior time step, grid-block pressures at the prior and next time steps, time-step size, grid-block size, well and outer boundary constraints, and rock and fluid properties. As noted in Appendix A above, these dependencies are used to define features and targets for artificial neural network algorithms, which serves as the basis for our problem formulation.

References

1. Metz, B.; Davidson, O.; Coninck, H. d.; Loos, M.; Meyer, L., *IPCC special report on carbon dioxide capture and storage*. Cambridge University Press, New York, NY (United States); Intergovernmental Panel on Climate Change, Geneva (Switzerland). Working Group III: 2005; p Medium: X; Size: 440; 23.3 MB pages.
2. Orr, F. M., Jr., Carbon Capture, Utilization, and Storage: An Update. *SPE Journal* **2018,** *23* (06), 2444-2455.
3. Page, B.; Turan, G.; Alex, Z.; Burrows, J.; Consoli, C.; Erikson, J.; Havercroft, I.; Kearns, D.; Liu, H.; Rassool, D.; Tamme, E.; Temple-Smith, L.; Townsend, A.; Zhang, T. *Global Status of CCS Report 2020*; 2020.
4. Greenberg, S.; Gauvreau, L.; Hnottavange-Telleen, K.; Finley, R.; Marsteller, S., Meeting CCS communication challenges head-on: Integrating communications, planning, risk assessment, and project management. *Energy Procedia* **2011,** *4*, 6188-6193.
5. Leetaru, K. H.; Leetaru, H. E., A Global Big Data Assessment of Public Attitudes Towards CCS Through the Media. *Energy Procedia* **2014,** *63*, 7011-7018.
6. Nordbotten, J.; Celia, M., Geological Storage of CO: Modeling Approaches for Large-Scale Simulation. *Geological Storage of CO2: Modeling Approaches for Large-Scale Simulation* **2011**, i-ix.
7. Pruess, K.; García, J.; Kovscek, T.; Oldenburg, C.; Rutqvist, J.; Steefel, C.; Xu, T., Code intercomparison builds confidence in numerical simulation models for geologic disposal of CO2. *Energy* **2004,** *29* (9), 1431-1444.
8. Class, H.; Ebigbo, A.; Helmig, R.; Dahle, H. K.; Nordbotten, J. M.; Celia, M. A.; Audigane, P.; Darcis, M.; Ennis-King, J.; Fan, Y.; Flemisch, B.; Gasda, S. E.; Jin, M.; Krug, S.; Labregere, D.; Naderi Beni, A.; Pawar, R. J.; Sbai, A.; Thomas, S. G.; Trenty, L.; Wei, L., A benchmark study on problems related to CO2 storage in geologic formations. *Computational Geosciences* **2009,** *13* (4), 409.

9. Sefat, M. H.; Salahshoor, K.; Jamialahmadi, M.; Moradi, B., A New Approach for the Development of Fast-analysis Proxies for Petroleum Reservoir Simulation. *Pet. Sci. Technol.* **2012,** *30* (18), 1920-1930.
10. Schuetter, J.; Mishra, k. S.; Ganesh, P. R.; Mooney, D., Building Statistical Proxy Models for CO2 Geologic Sequestration. *Energy Procedia* **2014,** *63*, 3702-3714.
11. Kalantari-Dahaghi, A.; Mohaghegh, S.; Esmaili, S., Data-driven proxy at hydraulic fracture cluster level: A technique for efficient CO2- enhanced gas recovery and storage assessment in shale reservoir. *Journal of Natural Gas Science and Engineering* **2015,** *27*, 515-530.
12. Singh, H., Machine learning for surveillance of fluid leakage from reservoir using only injection rates and bottomhole pressures. *Journal of Natural Gas Science and Engineering* **2019,** *69*, 102933.
13. Zhu, Y.; Zabaras, N., Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. *Journal of Computational Physics* **2018,** *366*, 415-447.
14. Raissi, M.; Perdikaris, P.; Karniadakis, G. E., Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. *arXiv e-prints* **2017**, arXiv:1711.10561.
15. Raissi, M.; Perdikaris, P.; Karniadakis, G. E., Physics Informed Deep Learning (Part II): Data-driven Discovery of Nonlinear Partial Differential Equations. *arXiv e-prints* **2017**, arXiv:1711.10566.
16. Géron, A. l., *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems*. O'Reilly Media: Sebastopol, CA, 2017.
17. Alakeely, A.; Horne, R. N., Simulating the Behavior of Reservoirs with Convolutional and Recurrent Neural Networks. *SPE Reservoir Evaluation & Engineering* **2020,** *23* (03), 0992-1005.
18. Ghassemzadeh, S.; Gonzalez Perdomo, M.; Haghighi, M.; Abbasnejad, E., A data-driven reservoir simulation for natural gas reservoirs. *Neural Computing and Applications* **2021**.
19. Golzari, A.; Haghighat Sefat, M.; Jamshidi, S., Development of an adaptive surrogate model for production optimization. *Journal of Petroleum Science and Engineering* **2015,** *133*, 677-688.
20. Tang, M.; Liu, Y.; Durlofsky, L. J., A deep-learning-based surrogate model for data assimilation in dynamic subsurface flow problems. *J. Comput. Phys.* **2020,** *413*, 109456.
21. Jin, Z. L.; Liu, Y.; Durlofsky, L. J., Deep-learning-based surrogate model for reservoir simulation with time-varying well controls. *Journal of Petroleum Science and Engineering* **2020,** *192*, 107273.
22. Zhong, Z.; Sun, A. Y.; Yang, Q.; Ouyang, Q., A deep learning approach to anomaly detection in geological carbon sequestration sites using pressure measurements. *Journal of Hydrology* **2019,** *573*, 885-894.
23. 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), 703-728.
24. Zhou, Z.; Lin, Y.; Zhang, Z.; Wu, Y.; Wang, Z.; Dilmore, R.; Guthrie, G., A data-driven CO2 leakage detection using seismic data and spatial–temporal densely connected convolutional neural networks. *International Journal of Greenhouse Gas Control* **2019,** *90*, 102790.
25. Zhong, Z.; Sun, A. Y.; Jeong, H., Predicting CO2 Plume Migration in Heterogeneous Formations Using Conditional Deep Convolutional Generative Adversarial Network. *Water Resour. Res.* **2019,** *55* (7), 5830-5851.
26. Wen, G.; Tang, M.; Benson, S. M., Towards a predictor for CO2 plume migration using deep neural networks. *International Journal of Greenhouse Gas Control* **2021,** *105*, 103223.
27. Song, Y.; Sung, W.; Jang, Y.; Jung, W., Application of an artificial neural network in predicting the effectiveness of trapping mechanisms on CO2 sequestration in saline aquifers. *International Journal of Greenhouse Gas Control* **2020,** *98*, 103042.
28. CMG-GEM, *GEM Technical Manual: General Adaptive Implicit Equation of State Compositional Model*. Computer Modelling Group: 1993.

29. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G. S.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Goodfellow, I.; Harp, A.; Irving, G.; Isard, M.; Jia, Y.; Jozefowicz, R.; Kaiser, L.; Kudlur, M.; Levenberg, J.; Mane, D.; Monga, R.; Moore, S.; Murray, D.; Olah, C.; Schuster, M.; Shlens, J.; Steiner, B.; Sutskever, I.; Talwar, K.; Tucker, P.; Vanhoucke, V.; Vasudevan, V.; Viegas, F.; Vinyals, O.; Warden, P.; Wattenberg, M.; Wicke, M.; Yu, Y.; Zheng, X., TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv e-prints* **2016**, arXiv:1603.04467.
30. Sze, V.; Chen, Y.; Yang, T.; Emer, J. S., Efficient Processing of Deep Neural Networks: A Tutorial and Survey. *Proceedings of the IEEE* **2017,** *105* (12), 2295-2329.
31. Wang, Z.; Bovik, A. C., Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures. *IEEE Signal Processing Magazine* **2009,** *26* (1), 98-117.
32. Zhou, W.; Bovik, A. C.; Sheikh, H. R.; Simoncelli, E. P., Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing* **2004,** *13* (4), 600-612.

1. Metz, B.; Davidson, O.; Coninck, H. d.; Loos, M.; Meyer, L., *IPCC special report on carbon dioxide capture and storage*. Cambridge University Press, New York, NY (United States); Intergovernmental Panel on Climate Change, Geneva (Switzerland). Working Group III: 2005; p Medium: X; Size: 440; 23.3 MB pages.
2. Orr, F. M., Jr., Carbon Capture, Utilization, and Storage: An Update. *SPE Journal* **2018,** *23* (06), 2444-2455.
3. Page, B.; Turan, G.; Alex, Z.; Burrows, J.; Consoli, C.; Erikson, J.; Havercroft, I.; Kearns, D.; Liu, H.; Rassool, D.; Tamme, E.; Temple-Smith, L.; Townsend, A.; Zhang, T. *Global Status of CCS Report 2020*; 2020.
4. Greenberg, S.; Gauvreau, L.; Hnottavange-Telleen, K.; Finley, R.; Marsteller, S., Meeting CCS communication challenges head-on: Integrating communications, planning, risk assessment, and project management. *Energy Procedia* **2011,** *4*, 6188-6193.
5. Leetaru, K. H.; Leetaru, H. E., A Global Big Data Assessment of Public Attitudes Towards CCS Through the Media. *Energy Procedia* **2014,** *63*, 7011-7018.
6. Nordbotten, J.; Celia, M., Geological Storage of CO: Modeling Approaches for Large-Scale Simulation. *Geological Storage of CO2: Modeling Approaches for Large-Scale Simulation* **2011**, i-ix.

7. Pruess, K.; García, J.; Kovscek, T.; Oldenburg, C.; Rutqvist, J.; Steefel, C.; Xu, T., Code intercomparison builds confidence in numerical simulation models for geologic disposal of CO2. *Energy* **2004,** *29* (9), 1431-1444.

8. Class, H.; Ebigbo, A.; Helmig, R.; Dahle, H. K.; Nordbotten, J. M.; Celia, M. A.; Audigane, P.; Darcis, M.; Ennis-King, J.; Fan, Y.; Flemisch, B.; Gasda, S. E.; Jin, M.; Krug, S.; Labregere, D.; Naderi Beni, A.; Pawar, R. J.; Sbai, A.; Thomas, S. G.; Trenty, L.; Wei, L., A benchmark study on problems related to CO2 storage in geologic formations. *Computational Geosciences* **2009,** *13* (4), 409.

9. Sefat, M. H.; Salahshoor, K.; Jamialahmadi, M.; Moradi, B., A New Approach for the Development of Fast-analysis Proxies for Petroleum Reservoir Simulation. *Pet. Sci. Technol.* **2012,** *30* (18), 1920-1930.

10. Schuetter, J.; Mishra, k. S.; Ganesh, P. R.; Mooney, D., Building Statistical Proxy Models for CO2 Geologic Sequestration. *Energy Procedia* **2014,** *63*, 3702-3714.

11. Kalantari-Dahaghi, A.; Mohaghegh, S.; Esmaili, S., Data-driven proxy at hydraulic fracture cluster level: A technique for efficient CO2- enhanced gas recovery and storage assessment in shale reservoir. *Journal of Natural Gas Science and Engineering* **2015,** *27*, 515-530.

12. Singh, H., Machine learning for surveillance of fluid leakage from reservoir using only injection rates and bottomhole pressures. *Journal of Natural Gas Science and Engineering* **2019,** *69*, 102933.

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

14. Raissi, M.; Perdikaris, P.; Karniadakis, G. E., Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. *arXiv e-prints* **2017**, arXiv:1711.10561.

15. Raissi, M.; Perdikaris, P.; Karniadakis, G. E., Physics Informed Deep Learning (Part II): Data-driven Discovery of Nonlinear Partial Differential Equations. *arXiv e-prints* **2017**, arXiv:1711.10566.

16. Géron, A. l., *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems*. O'Reilly Media: Sebastopol, CA, 2017.

17. Alakeely, A.; Horne, R. N., Simulating the Behavior of Reservoirs with Convolutional and Recurrent Neural Networks. *SPE Reservoir Evaluation & Engineering* **2020,** *23* (03), 0992-1005.

18. Ghassemzadeh, S.; Gonzalez Perdomo, M.; Haghighi, M.; Abbasnejad, E., A data-driven reservoir simulation for natural gas reservoirs. *Neural Computing and Applications* **2021**.

19. Golzari, A.; Haghighat Sefat, M.; Jamshidi, S., Development of an adaptive surrogate model for production optimization. *Journal of Petroleum Science and Engineering* **2015,** *133*, 677-688.

20. Tang, M.; Liu, Y.; Durlofsky, L. J., A deep-learning-based surrogate model for data assimilation in dynamic subsurface flow problems. *J. Comput. Phys.* **2020,** *413*, 109456.

21. Jin, Z. L.; Liu, Y.; Durlofsky, L. J., Deep-learning-based surrogate model for reservoir simulation with time-varying well controls. *Journal of Petroleum Science and Engineering* **2020,** *192*, 107273.

22. Zhong, Z.; Sun, A. Y.; Yang, Q.; Ouyang, Q., A deep learning approach to anomaly detection in geological carbon sequestration sites using pressure measurements. *Journal of Hydrology* **2019,** *573*, 885-894.

23. 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), 703-728.

24. Zhou, Z.; Lin, Y.; Zhang, Z.; Wu, Y.; Wang, Z.; Dilmore, R.; Guthrie, G., A data-driven CO2 leakage detection using seismic data and spatial–temporal densely connected convolutional neural networks. *International Journal of Greenhouse Gas Control* **2019,** *90*, 102790.

25. Zhong, Z.; Sun, A. Y.; Jeong, H., Predicting CO2 Plume Migration in Heterogeneous Formations Using Conditional Deep Convolutional Generative Adversarial Network. *Water Resour. Res.* **2019,** *55* (7), 5830-5851.

26. Wen, G.; Tang, M.; Benson, S. M., Towards a predictor for CO2 plume migration using deep neural networks. *International Journal of Greenhouse Gas Control* **2021,** *105*, 103223.

27. Song, Y.; Sung, W.; Jang, Y.; Jung, W., Application of an artificial neural network in predicting the effectiveness of trapping mechanisms on CO2 sequestration in saline aquifers. *International Journal of Greenhouse Gas Control* **2020,** *98*, 103042.

28. CMG-GEM, *GEM Technical Manual: General Adaptive Implicit Equation of State Compositional Model*. Computer Modelling Group: 1993.

29. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G. S.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Goodfellow, I.; Harp, A.; Irving, G.; Isard, M.; Jia, Y.; Jozefowicz, R.; Kaiser, L.; Kudlur, M.; Levenberg, J.; Mane, D.; Monga, R.; Moore, S.; Murray, D.; Olah, C.; Schuster, M.; Shlens, J.; Steiner, B.; Sutskever, I.; Talwar, K.; Tucker, P.; Vanhoucke, V.; Vasudevan, V.; Viegas, F.; Vinyals, O.; Warden, P.; Wattenberg, M.; Wicke, M.; Yu, Y.; Zheng, X., TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv e-prints* **2016**, arXiv:1603.04467.

30. Sze, V.; Chen, Y.; Yang, T.; Emer, J. S., Efficient Processing of Deep Neural Networks: A Tutorial and Survey. *Proceedings of the IEEE* **2017,** *105* (12), 2295-2329.

31. Wang, Z.; Bovik, A. C., Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures. *IEEE Signal Processing Magazine* **2009,** *26* (1), 98-117.

32. Zhou, W.; Bovik, A. C.; Sheikh, H. R.; Simoncelli, E. P., Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing* **2004,** *13* (4), 600-612.