

SANDIA REPORT

SAND2021- 11262

Printed September 2021

**Sandia
National
Laboratories**

Predictive Data-driven Platform for Subsurface Energy Production

Hongkyu Yoon, Stephen J Verzi, Katherine Cauthen, Srideep Musuvathy, Darryl Melander, Kyle T. Norland, Adriana M Morales, Jonghyun Lee, and Alexander Y. Sun

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico
87185 and Livermore,
California 94550

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology & Engineering Solutions of Sandia, LLC.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from

U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@osti.gov
Online ordering: <http://www.osti.gov/scitech>

Available to the public from

U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Rd
Alexandria, VA 22312

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.gov
Online order: <https://classic.ntis.gov/help/order-methods/>



ABSTRACT

Subsurface energy activities such as unconventional resource recovery, enhanced geothermal energy systems, and geologic carbon storage require fast and reliable methods to account for complex, multiphysical processes in heterogeneous fractured and porous media. Although reservoir simulation is considered the industry standard for simulating these subsurface systems with injection and/or extraction operations, reservoir simulation requires spatio-temporal “Big Data” into the simulation model, which is typically a major challenge during model development and computational phase. In this work, we developed and applied various deep neural network-based approaches to (1) process multiscale image segmentation, (2) generate ensemble members of drainage networks, flow channels, and porous media using deep convolutional generative adversarial network, (3) construct multiple hybrid neural networks such as convolutional LSTM and convolutional neural network-LSTM to develop fast and accurate reduced order models for shale gas extraction, and (4) physics-informed neural network and deep Q-learning for flow and energy production. We hypothesized that physics-based machine learning/deep learning can overcome the shortcomings of traditional machine learning methods where data-driven models have faltered beyond the data and physical conditions used for training and validation. We improved and developed novel approaches to demonstrate that physics-based ML can allow us to incorporate physical constraints (e.g., scientific domain knowledge) into ML framework. Outcomes of this project will be readily applicable for many energy and national security problems that are particularly defined by multiscale features and network systems.

ACKNOWLEDGEMENTS

This work was supported by the Sandia National Laboratories Laboratory Directed Research and Development program, Project No. 213008. This work was originally developed with former team members including Katherine Klise, Anneliese Lilje, and Brad Aimone who contributed to developing the original scope of the project. We appreciate Warren Davis for his continuing participation in regular project meetings, Jack Ringer and Vincent Liu (undergraduate interns) for semantic segmentation and physics-informed neural network, Adriana M. Miranda and Kyle T. Norland (summer interns) for deep reinforcement learning development, Jamshed Kaikaus (summer intern) for CNN-LSTM development, Sangcheol Yoon (graduate student at Texas A&M) for coupled reservoir simulations, and Dr. Sung Eun Kim (The Seoul Institute) for GAN-based image generation works. We also thank Moo Lee (retired) for his advice and support with project management.

CONTENTS

1. Introduction.....	11
2. Porous Materials for Subsurface Resources.....	12
2.1. Image Segmentation of Multiscale Porous Media.....	13
2.2. Pore Network Generation Using Generative Adversarial Networks.....	15
2.3. Permeability Estimation of Porous Media.....	16
3. Unconventional Resources Recovery.....	18
3.1. Shale Gas Extraction.....	18
3.2. Multiphysics of Shale Resource Recovery.....	18
4. Machine Learning Models for Shale Gas Recovery.....	20
4.1. Description of Machine Learning Architectures.....	20
4.1.1. LSTM.....	20
4.1.2. ConvLSTM.....	20
4.1.3. CNN-LSTM.....	21
4.2. Data Processing.....	21
4.3. Results.....	21
4.3.1. ConvLSTM.....	21
4.3.2. CNN-LSTM.....	23
5. Reinforcement Learning of Operation.....	25
5.1. RL Introduction.....	25
5.1.1. Q-learning.....	26
5.1.2. Deep Q-learning.....	26
5.1.3. Target network.....	26
5.1.4. Experience replay.....	27
5.1.5. Policy and policy gradients.....	27
5.1.6. Reinforce with baseline.....	27
5.1.7. Trust region policy optimization (TRPO).....	27
5.1.8. Proximal policy optimization (PPO).....	28
5.2. Model Description.....	28
5.2.1. Problem setup.....	28
5.2.2. Action space.....	28
5.2.3. Reward design.....	29
5.2.4. The environment.....	29
6. Physics Informed Neural Networks.....	30
6.1. PINN Introduction.....	30
6.1.1. Burger's equation.....	31
6.2. Stokes' Flow Equation.....	32
6.2.1. PINN training.....	33
7. Conclusions.....	35

LIST OF FIGURES

Figure 2-1. Examples of multiscale pore network systems of glass beads (microCT), sandstone (microCT), carbonate chalk (FIB-SEM, from Yoon and Dewers (2013)), and Marcellus shale nanopore (FIB-SEM). Binary images show 3D segmented images over small regions and pore network (PN) extracted from segmented images and images generated with (deep convolutional) generative adversarial networks (DCGAN) are also shown.	12
Figure 2-2. Examples of multiscale imaging of Marcellus shale. (a) large scale BSE image (2 μm resolution, 1 inch diameter), (b) BSE image at 0.2 μm resolution (80 μm x60 μm), and (c-d) TEM images at two different resolutions (see the scale bar on the figure). Image in (d) clearly shows nano-pores within organic areas. Light and dark gray colors represent pore and organic phases, respectively. Area for FIB-SEM in (b) is used for 3D FIB-SEM imaging of shale used in Figure 2-3.	13
Figure 2-3. Semantic segmentation results using four different deep learning methods including U-Net, 3D U-Net, U-VGG16, and U-Resnet. Four different rock samples are used. Label and original image and ML-prediction of segmentation are shown. (a-c) binary label is used to represent pore and solid matrix, while (d) shale has three different phases including pore, organic, and solid matrix phases. On the error maps, blue and red colors represent false positive and true negative cases, respectively.	14
Figure 2-4. Image generation using SAGANs for (a) beadpack and (b) channel pattern. Each segment size with a dashed box was smaller than the final image generated. The figures are from Kim et al. (2021b).	15
Figure 2-5. Schematic of CNN and MLP architecture for permeability estimation. Outputs from CNN and MLP models are concatenated to develop a regression model of permeability prediction. The numbers in different colors show sensitivity cases. The model architecture was modified from Wu et al. [2018].	16
Figure 2-6. Comparison of the permeability prediction with three different models. The linear regression fitting is also shown. The black dot line represents the perfect predicted case. Modified from Yoon et al. [2020].	17
Figure 3-1. (a) Numerical mesh and horizontal well with hydraulic fractures, (b) pressure distribution at the end of simulation (t=10958.07 days), and (c) example of production curve over time. Simulations were performed using MRST-Shale [Wang, 2020]	18
Figure 3-2. (a) Workflow of simulations for unconventional resource recovery and (b) cumulative density function (CDF) as a function of fracture length generated by hydraulic fracturing. Four different cases are considered to evaluate the impact of child wells on fracture length prediction.	19
Figure 4-1. (a) ConvLSTM model for cumulative production and (b) ConvLSTM model for pressure distribution	21
Figure 4-2. Actual and ConvLSTM predicted values of cumulative production for three test cases.	22
Figure 4-3. Actual and ConvLSTM predicted values of the pressure distribution for three test cases at three timepoints.	22
Figure 4-4. (a) CNN-LSTM model for cumulative production and (b) CNN-LSTM model for pressure distribution.	23
Figure 4-5. Actual and CNN-LSTM predicted values of cumulative production for three test cases.	24
Figure 4-6. Actual and CNN-LSTM predicted values of the pressure distribution for three test cases at three timepoints.	24

Figure 5-1. (a) The schematic of agent-environment interaction in reinforcement learning (redrawn from Sutton and Barto [2018]) and (b) the three components of the DRL agent in this work: Deep Q-Network (DQN), Target Network, and Experience Memory.	25
Figure 5-2. Diagram of deep reinforcement learning framework.....	28
Figure 6-1. Schematic of PINN architecture (modified from Rossi et al., 2019).....	31
Figure 6-2. (a) An example of solution to Burger’s equation over $u(x,t)$ and (b-c) comparison of true solution and PINN prediction. The domain size is 500 (x) x 200 (t) points.....	31
Figure 6-3. (a) Numerical domain, single cylinder obstacle, and flow direction of the stokes flow simulation, and (b) staircase discretization of the cylinder obstacle with zero velocity boundary conditions normal to the face of boundary.....	32
Figure 6-4. Stokes flow simulation in the presence of single cylinder obstacle. Solid line represents streamlines.....	33
Figure 6-5. Actual and ML predicted velocities with residual difference along the x and y directions.	34
Figure 6-6. Actual and ML predicted pressure filed.	34

LIST OF TABLES

Table 4-1. ConvLSTM hyperparameters	22
Table 4-2. CNN-LSTM hyperparameters	24

This page left blank

ACRONYMS AND DEFINITIONS

Abbreviation	Definition
CNN	Convolutional Neural Network
LSTM	Long Short-Term Memory
ML	Machine Learning
DL	Deep Learning
ConvLSTM	Convolutional LSTM
PINN	Physics-Informed Neural Network
3D	Three-dimensional
microCT	Micro-Computed tomography
FIB-SEM	Focused Ion Beam-Scanning Electron Microscopy
GAN	Generative Adversarial Network
DCGAN	Deep Convolutional Generative Adversarial Network
BSE	Backscattered Scanning Electron Microscopy
TEM	Transmission Electron Microscopy
SAGAN	Spatially Assembled Generative Adversarial Network
MRST	MATLAB Reservoir Simulation Toolbox
RNN	Recurrent Neural Network
DRL	Deep Reinforcement Learning
RL	Reinforcement Learning
MDP	Markov Decision Process
DQN	Deep Q-Network
DNNs	Deep Neural Networks
TRPO	Trust Region Policy Optimization
PPO	Proximal Policy Optimization

This page left blank

1. INTRODUCTION

Shale reservoirs are characterized by low permeability rocks that are highly heterogeneous over relatively small length scales and have low connectivity between natural fractures. Hydraulic fracturing aims to increase permeability by connecting natural flow pathways. Hence, subsurface energy production is governed by complex, multiscale processes. To predict reservoir production, this complex network, along with flow regimes that govern advection, diffusion, and adsorption must be characterized. Traditional reservoir simulation includes modeling complex fracture sets along with flow and transport [e.g., Salama et al., 2017]. However, this methodology can be computationally expensive due to the large quantities of spatio-temporal data imbedded in the model.

Recent big successes in machine learning (ML) for both commercial and scientific & engineering domains have been attributed to advances in ML algorithms, innovation in computational hardware, and large amounts of data [e.g., LeCun et al., 2015]. Particularly deep learning (DL) methods have been used to efficiently extract features, discover knowledge, and identify correlations and patterns. Recent efforts to apply ML driven models for flow, transport, and coupled thermal-chemical-mechanical processes in natural and engineered environments demonstrated promising outcomes [e.g., Ling et al., 2016; Kadeethum et al., under review]. Numerical simulations, governing equations, physical constraints, laboratory observations, and field measurement data all can be utilized to develop data-driven models in the physics-informed ML framework. These recent advances in ML algorithms and rapid development in sensing techniques have provided great opportunities to make a drastic improvement in the computational efficiency and realtime forecasting both fundamentally and practically. Our recent survey on “Applications of physics-informed scientific machine learning in subsurface science: A survey” provides an extensive review on the recent development of science/physics guided machine learning methods and applications in subsurface reservoir systems [Sun et al., 2021].

In this project we developed a predictive data-driven platform that offers a set of algorithms including various ML/DL methods by incorporating physics-based knowledge into ML, while attempting to answer the following questions: How do we incorporate known constraints (e.g., physics, geological and operational conditions) to improve the accuracy and confidence of ML predictions? How do we reliably train DL tools and expand the trained model for long-term prediction? How can we retrain the model once systems will be changed? As highlighted in a recent workshop on the scientific machine learning for science and engineering [DOE ASCR, 2018], our approach is to advance the data-driven model reduction methods using physics-based ML for dynamical systems (e.g., production decline analysis).

The rest of the report is summarized as followings. First, multiscale images of a representative Marcellus shale sample image are presented with semantic segmentation of various three-dimensional (3D) rock images using multiple deep learning algorithms and permeability estimation using deep learning methods in Section 2. Description of reservoir simulation results is reported in Section 3. Performance of ML-driven model against reservoir simulation results is described in Section 4, followed by the outline of optimal well operation using reinforcement learning method in Section 5. Finally, we provide examples of physics-informed neural networks (PINNs) in Section 6 before the conclusions.

2. POROUS MATERIALS FOR SUBSURFACE RESOURCES

We will build comprehensive reduced order models of unconventional resources recovery. The first step to model subsurface energy recovery is to understand a porous media system. Natural porous media consist of multiple features of pore characteristics (e.g., shape, sizes, degree of connectivity) which are often scale- and material-dependent. Representative 3D rock images obtained from micro-computed tomography (microCT) and focused ion beam-scanning electron microscopy (FIB-SEM) are shown in Figure 2-1. As shown, scales of pore networks and connectivity vary depending on materials. For quantitative modeling of flow and transport in fractured and porous media, the first step is to segment images, followed by pore network generation which was performed using open source pore network model, OpenPNM [Gostick et al., 2016]. Pore networks can represent multiscale connections from nanopores to natural fractures to induced fractures in order to reflect multiscale pathways in simplified architectures during unconventional production (e.g., hydraulic fracturing). For example, carbonate chalk image was used to construct a physical microfluidic device to study the impact of salinity to evaluate residual oil recovery [Park et al., 2021]. One key aspect is to account for stochastic nature of porous media systems, which has been demonstrated in new machine learning framework using generative adversarial networks (GANs) [e.g., Kim et al., 2021a & 2021b].

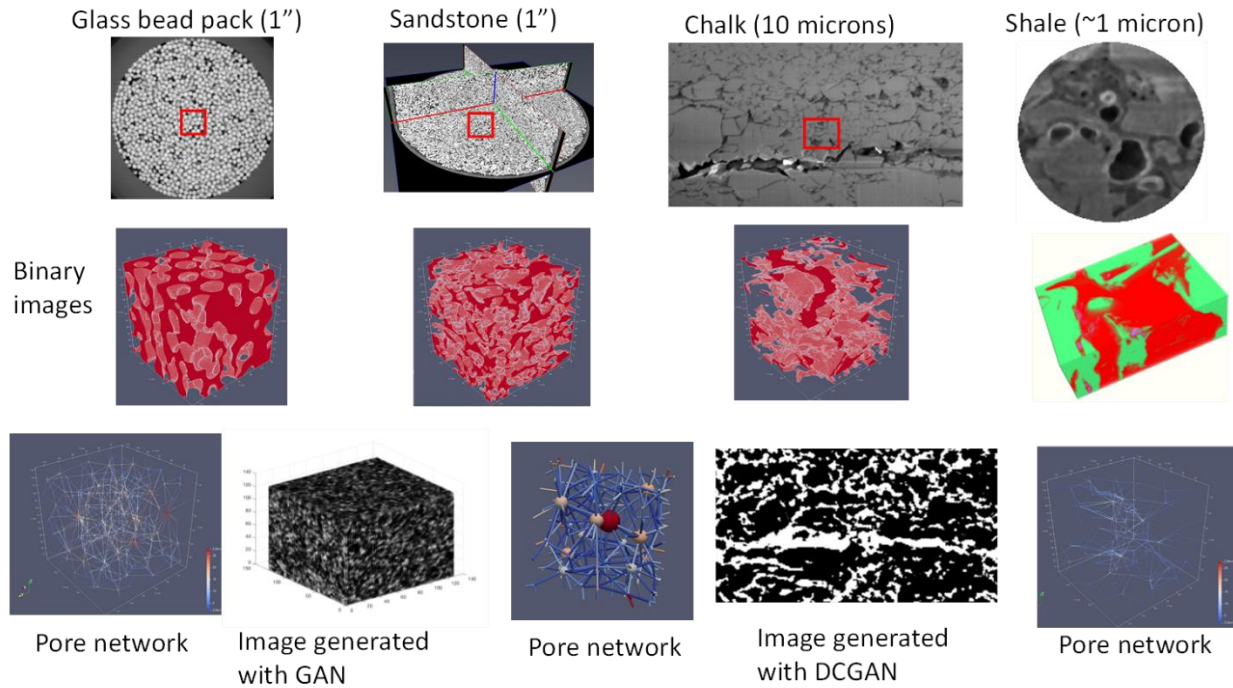


Figure 2-1. Examples of multiscale pore network systems of glass beads (microCT), sandstone (microCT), carbonate chalk (FIB-SEM, from Yoon and Dewers (2013)), and Marcellus shale nanopore (FIB-SEM). Binary images show 3D segmented images over small regions and pore network (PN) extracted from segmented images and images generated with (deep convolutional) generative adversarial networks (DCGAN) are also shown.

2.1. Image Segmentation of Multiscale Porous Media

One example of multiscale porous media is shown in Figure 2-2 where multiscale backscattered scanning electron microscopy (BSE), FIB-SEM, and transmission electron microscopy (TEM) were used to characterize pore topology, structure, and composition. Although our imaging capabilities have been developed dramatically, the imaging analysis methods in the open source domain compared to commercial software have not been improved significantly. Consequence is to use relatively simple threshold-based technique very commonly. Hence, recent advances in ML/DL [LeCun et al., 2015] provide grand opportunities to improve the open-source framework for image analysis of digital rock images. In this work, multiple deep learning methods for semantic segmentation are used to evaluate various 3D images of earth rock materials. Briefly, three two dimensional (2D) models including U-Net [Ronneberger et al. 2015], VGG16 [Simonyan and Zisserman, 2014], and ResNet [He et al. 2016] and one 3D version of the U-Net are used. VGG16 and ResNet are used as encoder in the U-Net architecture. Hence, all models have the form of an encoder-decoder U-Net architecture.

The performance of three different 2D DL architectures as well as a 3D U-Net is assessed on four independent datasets. Each of these datasets is composed of images and corresponding ground truth labels. The training data is used to fit each model, validation to monitor training, and testing to evaluate model performance. Example of segmentation results is shown in Figure 2-3. These results indicate that DL models can successfully be applied to the task of semantic segmentation for all four rock images. Of note is that the qualitative performance of deep learning models is heavily

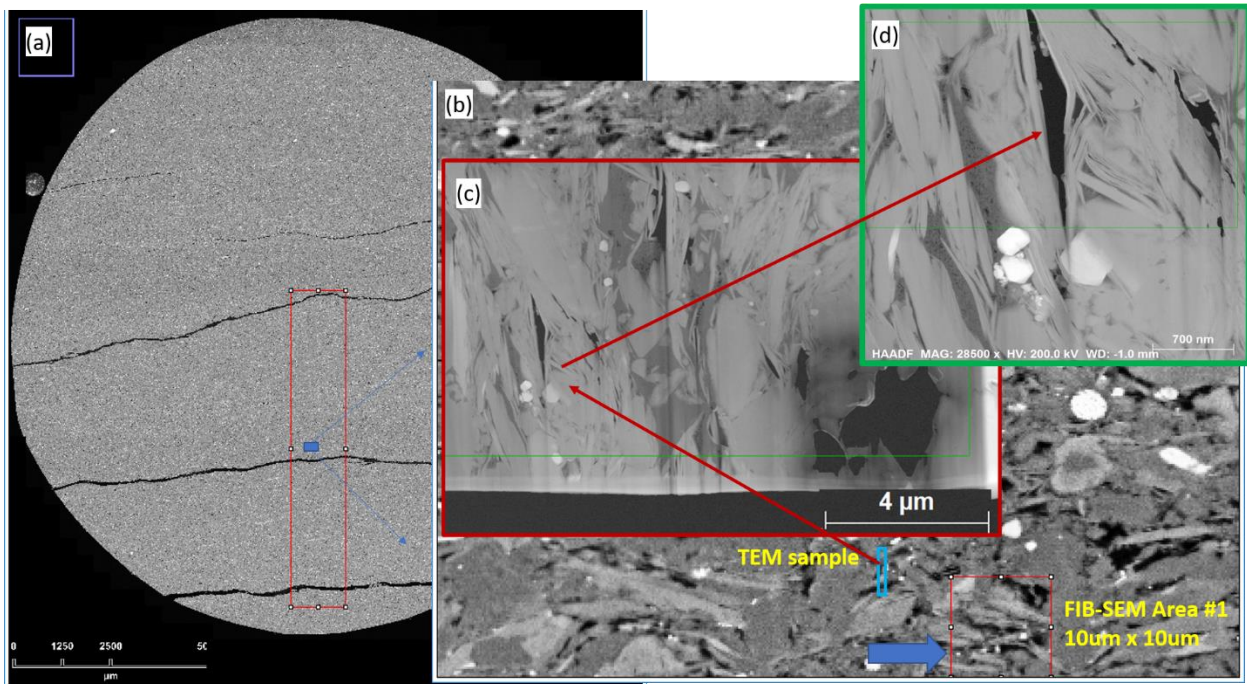


Figure 2-2. Examples of multiscale imaging of Marcellus shale. (a) large scale BSE image (2 μm resolution, 1 inch diameter), (b) BSE image at 0.2 μm resolution (80 μm x60 μm), and (c-d) TEM images at two different resolutions (see the scale bar on the figure). Image in (d) clearly shows nano-pores within organic areas. Light and dark gray colors represent pore and organic phases, respectively. Area for FIB-SEM in (b) is used for 3D FIB-SEM imaging of shale used in Figure 2-3.

dependent upon the quality of training data. Although models are depending on the quality of input data, we found that these DL methods can recover information not captured by original segmentation labels. The use of pretrained weights with ImageNet [Deng et al., 2009] was found to improve the accuracy of U-ResNet model after training with rock images, indicating that transfer learning can improve performance for these networks. U-VGG16 benefitted less from transfer learning, likely because the U-VGG16 has a simpler architecture compared to U-ResNet, making the model less robust with new training data. The 3D U-Net model was found to unexpectedly underperform in comparison to the 2D case. One explanation for this underperformance of the 3D case is a lack of training data. More comprehensive analysis and ensemble approach will be discussed in the manuscript [Yoon and Ringer, in prep].

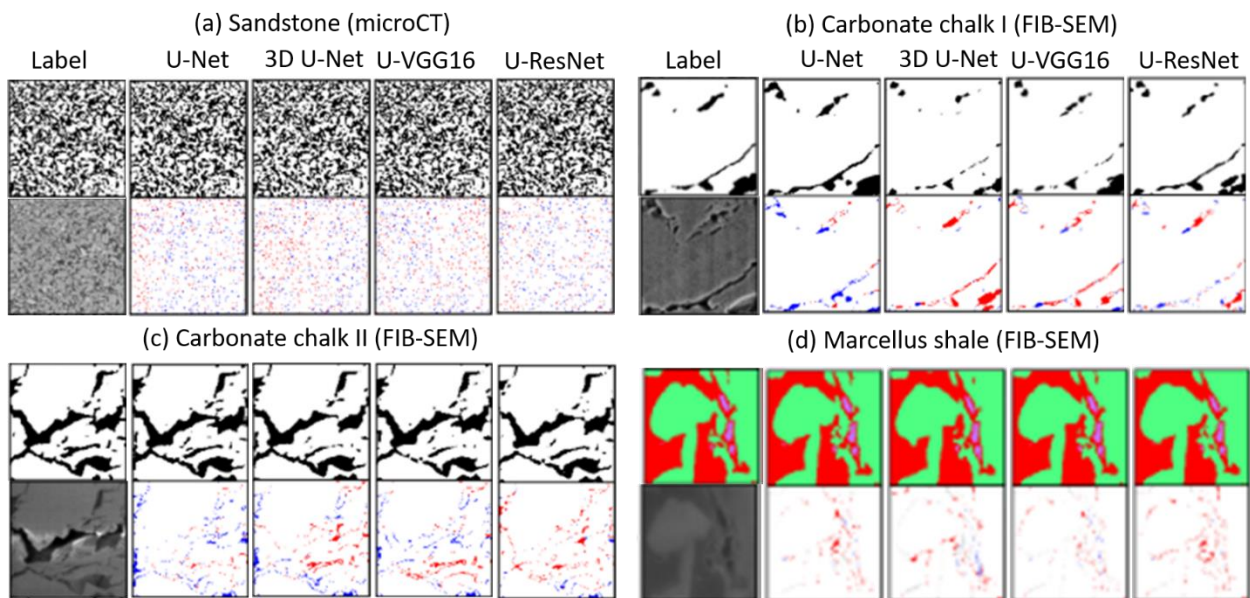


Figure 2-3. Semantic segmentation results using four different deep learning methods including U-Net, 3D U-Net, U-VGG16, and U-Resnet. Four different rock samples are used. Label and original image and ML-prediction of segmentation are shown. (a-c) binary label is used to represent pore and solid matrix, while (d) shale has three different phases including pore, organic, and solid matrix phases. On the error maps, blue and red colors represent false positive and true negative cases, respectively.

2.2. Pore Network Generation Using Generative Adversarial Networks

Deep generative models such as variational autoencoders (VAEs) and generative adversarial networks (GANs) can be used to form a low-dimensional latent variable representation of the data used in training. In nano-porous media considered in this work, coupled processes occur in heterogeneous porous media whose properties are partially observed and imperfectly collected in practice. Hence, available data is often noisy, corrupted, and/or sparse. This reveals the limitation of the deterministic prediction that does not account for the errors in data collection and process modeling and it is also a very challenging goal to make accurate and computationally feasible predictions over a range of uncertain parameter spaces.

Geomaterials with complex geometry and topology such as shales and carbonate rocks are typically characterized with sparse field data samples. Accordingly, generating arbitrary size of 3D rock structures with topological and connectivity features similar to original samples at a low computation cost is important to improve realistic geomaterial reconstruction. Recent progress in GAN-based image generation has demonstrated remarkable results over traditionally implemented statistical methods [e.g., Kim et al., 2021a]. However, GANs are often limited by considerable computational costs for high-dimensional image reconstruction applications. In this work, we developed a new method called the spatially assembled GANs (SAGANs) where a generator is trained to create multiple segments (like a mosaic) and then, spatial characteristics are evaluated by a discriminator to enhance the training of the generator so that each segment can be formed to a large scale image whose multiscale features are consistent with those in the original samples [Kim et al., 2021b]. The performance of the SAGANs was tested by reconstructing widely-used 2D and 3D geomaterials. It is shown that SAGANs can generate the arbitrary size of statistical realizations with similar connectivity and structural properties to training samples as shown in Figure 2-4. The computational time is significantly improved compared to those from standard GANs, especially for 3D geomaterial reconstruction examples. The detailed description of model architecture and uncertainty quantification of generated models is provided in Kim et al. [2021b].

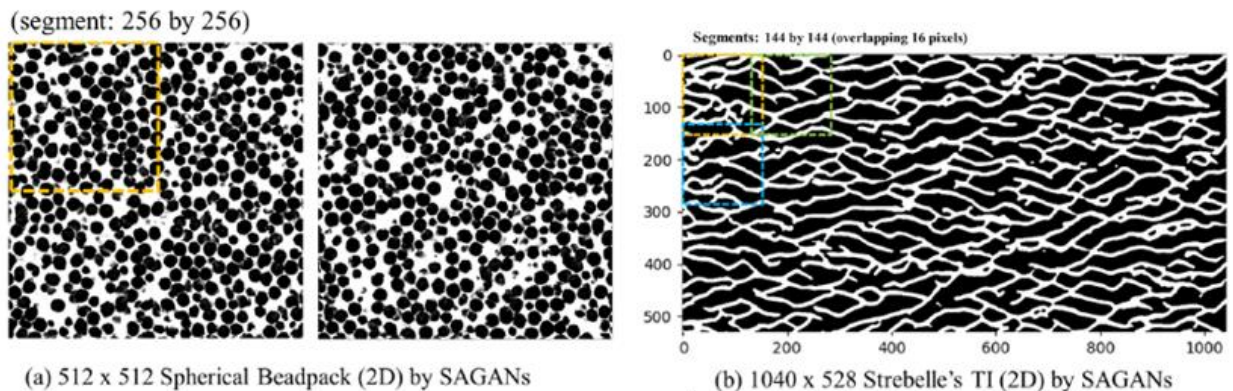


Figure 2-4. Image generation using SAGANs for (a) beadpack and (b) channel pattern. Each segment size with a dashed box was smaller than the final image generated. The figures are from Kim et al. (2021b).

2.3. Permeability Estimation of Porous Media

Flow and reactive transport processes in porous media is important to advance our understanding of subsurface energy recovery. One of key parameters is a permeability of porous media. Here we first used synthetic 2D porous media to estimate the permeability of porous media using 2D convolutional neural network (CNN) with constraints of physical properties (e.g., porosity and surface area) [Yoon et al., 2020] and carbonate chalk 3D porous media using 3D CNN architecture [Yoon et al., 2021]. Detailed description of CNN architectures, hyperparameters, input and output, and training procedures are provided in Yoon et al. [2020, 2021]. Here we briefly report key aspects of permeability estimation of porous media using CNN and MLP architecture as shown in Figure 2-5. Although CNN model is a standard 2D CNN architecture to extract features, a multilayer perception (MLP) model assimilates physical properties of each image (i.e., porosity and surface area) into a regression model by concatenating outputs from both CNN and MLP. Figure 2-5 shows key hyperparameters that are evaluated to obtain optimal values (through manual sensitivity analysis), while automatic hyperparameter optimization was used for 3D model architecture in Yoon et al. [2021]. The procedure of hyperparameter optimization is also described in the Section 4.

The architecture of CNN1-modified case is shown in Figure 2-5, which is modified from the model in Wu et al. [2018] where a similar approach with two numeric values was evaluated. As shown in Figure 2-6, both CNN-modified cases predict the permeability of test datasets better than the CNN2 with two numeric data. Hence, this comparison shows the importance of CNN architecture on model prediction. The CNN1 with image only performed similarly to the CNN2 model (used in Wu et al., 2018) with two numeric data. Comparison of the CNN1-modified cases shows that the case with MLP output has a slope value closer to the 1:1 line than the case with two numeric data, while the R^2 values are similar. Note that all ML model cases with two numerical data outperformed the CNN cases without additional data and with either one of porosity or surface area [see Yoon et al., 2020].

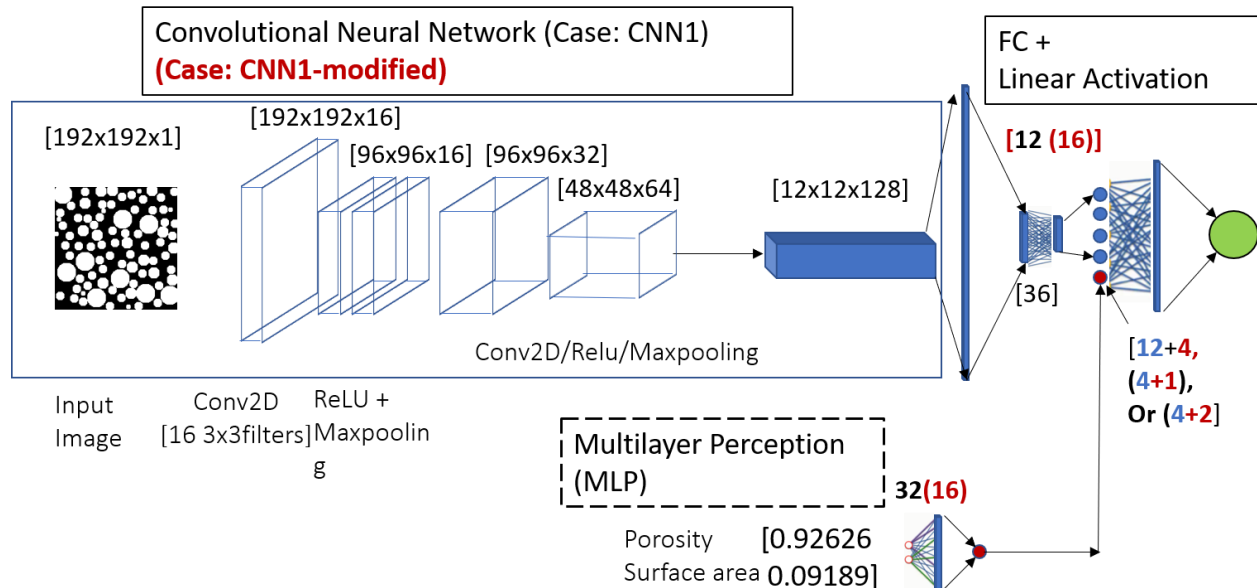


Figure 2-5. Schematic of CNN and MLP architecture for permeability estimation. Outputs from CNN and MLP models are concatenated to develop a regression model of permeability prediction. The numbers in different colors show sensitivity cases. The model architecture was modified from Wu et al. [2018].

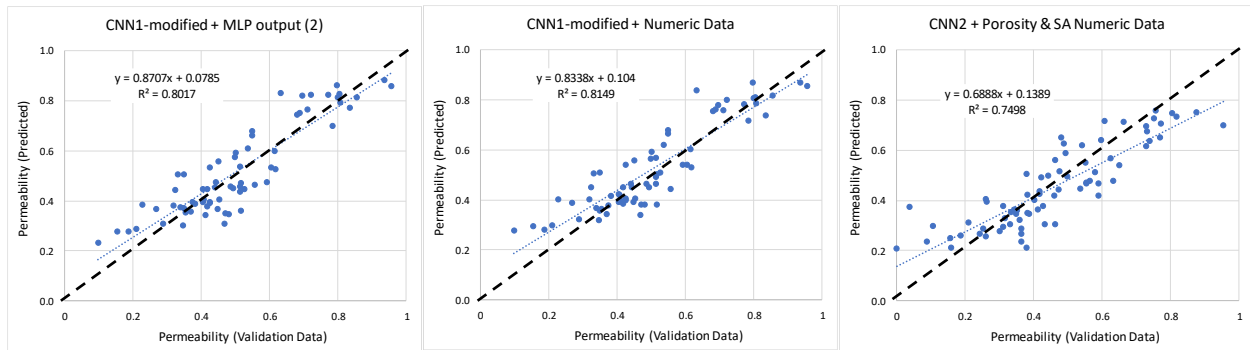


Figure 2-6. Comparison of the permeability prediction with three different models. The linear regression fitting is also shown. The black dot line represents the perfect predicted case. Modified from Yoon et al. [2020].

3. UNCONVENTIONAL RESOURCES RECOVERY

3.1. Shale Gas Extraction

We used an open source shale gas recovery model, MRST-Shale [Wang, 2000] to generate gas extraction data that was used to build ML-based models in Section 4. MRST-shale is developed based on the platform of the MATLAB Reservoir Simulation Toolbox (MRST) [Lie et al., 2012]. In particular, the fracture network system is solved using an embedded fracture network system. Key transport processes including gas adsorption/desorption, non-Darcy flow by apparent permeability (Knudsen diffusion, surface diffusion, slippage) are all embedded into a Darcy's equation of gas flow. Geomechanical effect on fractured networks was not considered in MRST-Shale. One simple example used in this work is shown in Figure 3-1. Numerical mesh with a horizontal well with multiple perforation points was used to simulate gas extraction. To generate training data, five different parameters including permeability of the matrix, porosity, fracture permeability, bottom hole pressure, and fracture aperture size are sampled uniformly. Output includes time series of production rate and pressure distribution change over the numerical domain (Figure 3-1b-c).

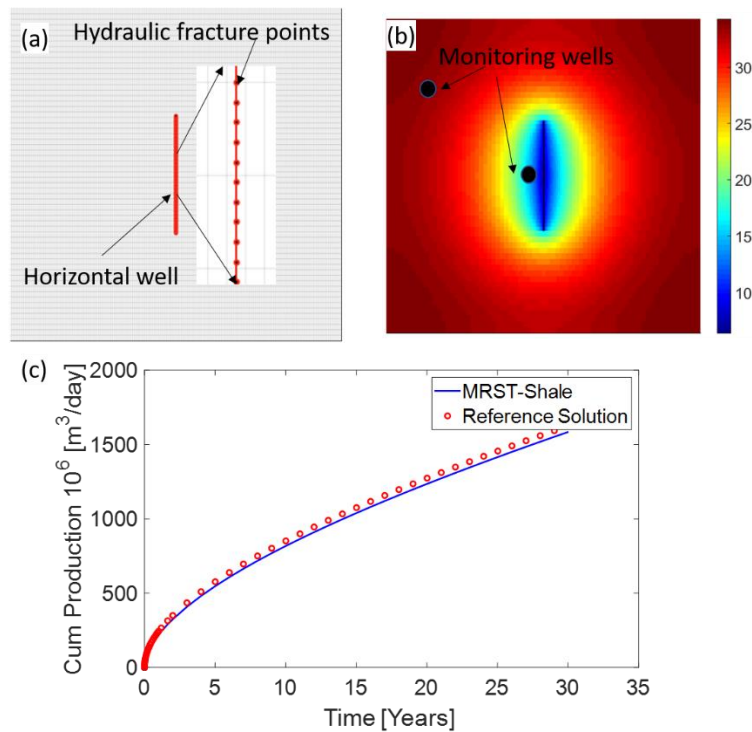


Figure 3-1. (a) Numerical mesh and horizontal well with hydraulic fractures, (b) pressure distribution at the end of simulation ($t=10958.07$ days), and (c) example of production curve over time. Simulations were performed using MRST-Shale [Wang, 2020]

3.2. Multiphysics of Shale Resource Recovery

Efficient and cost-effective unconventional oil and gas (UOG) recovery depends critically on the knowledge of primary factors controlling the reservoir producing behaviors, as well as on well completion strategies and well spacing. Completion efficiency is determined by stimulated reservoir volume, fracture size, and geometry among others. It is mainly controlled by the completion design

such as cluster spacing, clusters per stage, stage length, and pumping rate. The goal is to optimize engineering components to maximize the productivity of the single well, which can be achieved by training a machine-learning model to find the optimum time and distance of child wells. In this work, a full-workflow and code for optimization of the completion job using commercial software package (Kinetix, INTERSECT, and VISAGE) and its field application of optimization in completion strategy for both single well and multiple wells scenario are considered. (Figure 3-2a). The impact of well spacing and child well installation period on fracture length is shown in Figure 3-2b. The best case was found with 660-ft well spacing and child well installation at the same time as the parent wells. These simulation results will be used to develop ML-driven models in near future.

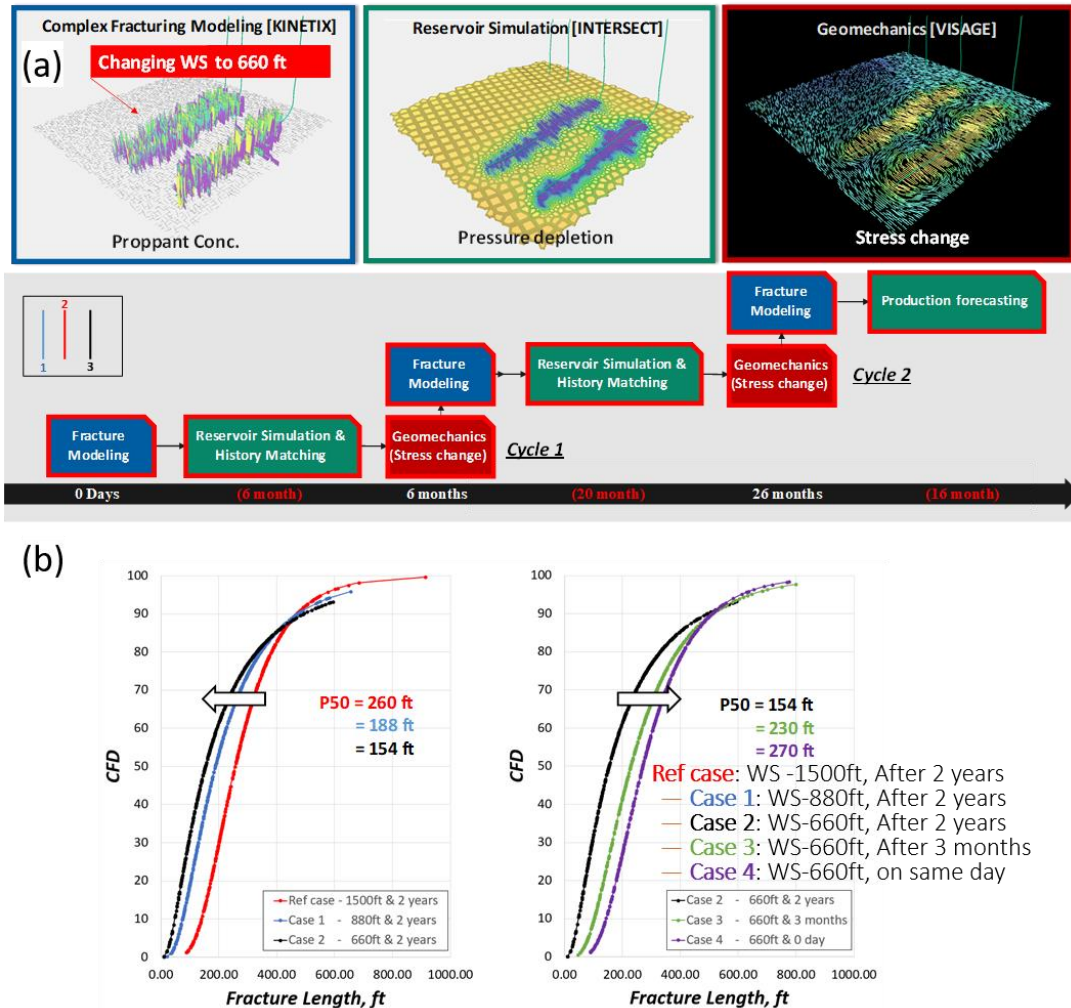


Figure 3-2. (a) Workflow of simulations for unconventional resource recovery and (b) cumulative density function (CDF) as a function of fracture length generated by hydraulic fracturing. Four different cases are considered to evaluate the impact of child wells on fracture length prediction.

4. MACHINE LEARNING MODELS FOR SHALE GAS RECOVERY

4.1. Description of Machine Learning Architectures

4.1.1. LSTM

A Long Short-Term Memory (LSTM) network is a type of recurrent neural network (RNN) that is able to learn long-term dependencies in time series data. One shortcoming of RNNs is that they suffer from the vanishing gradient problem, in which partial derivatives are calculated over the hidden layers during backpropagation and are then used by the gradient descent learning rule to adjust weights such that the error decreases [Bengio et al., 1994]. Since the derivative for a given layer is calculated using the chain rule across subsequent layers, when there are many time steps in the training data the gradient becomes very small for earlier layers. As a result, the weights of earlier layers are not effectively updated during training, leading to inaccuracy of the whole network. Unlike traditional RNNs, LSTMs allow information to be passed across layers relatively unchanged by using regulatory gates that learn which information is relevant to keep across time steps [Hochreiter and Schmidhuber, 1997]. Moreover, LSTMs remain relatively unperturbed if the amount of time between observations is unequal across the series.

A traditional LSTM model is defined below, where i_t, f_t, o_t are the gates, x_t are the input data, h_t are the hidden states, c_t are the cell states, and ‘ \circ ’ represents the element-wise product.

$$\begin{aligned}i_t &= \text{sigmoid}(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \\f_t &= \text{sigmoid}(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \\c_t &= f_t \circ c_{t-1} + i_t \circ \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\o_t &= \text{sigmoid}(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \\h_t &= o_t \circ \tanh(c_t)\end{aligned}$$

A traditional LSTM models temporal data, however, our data additionally have a spatial component. We implemented two different architectures which extend the LSTM network to accommodate spatiotemporal data, Convolutional LSTM (ConvLSTM) and CNN LSTM.

4.1.2. ConvLSTM

ConvLSTM is a variation on the traditional LSTM where, rather than performing matrix multiplication operations, the ConvLSTM cell applies a convolution operation in order to capture the spatial information contained in the data [Shi et al., 2015]. The convolution is done for both state-to-state and input-to-state transitions. Three-dimensional tensors, with time in the first dimension and space in the other two dimensions, are used for the input data X_1, \dots, X_t , hidden states H_1, \dots, H_t , cell states C_1, \dots, C_t , and gates i_t, f_t, o_t . The ConvLSTM model is defined by the following, where ‘ $*$ ’ represents convolution and ‘ \circ ’ represents the element-wise product.

$$\begin{aligned}i_t &= \text{sigmoid}(W_{xi} * X_t + W_{hi} * H_{t-1} + b_i) \\f_t &= \text{sigmoid}(W_{xf} * X_t + W_{hf} * H_{t-1} + b_f) \\C_t &= f_t \circ C_{t-1} + i_t \circ \tanh(W_{xc} * X_t + W_{hc} * H_{t-1} + b_c) \\o_t &= \text{sigmoid}(W_{xo} * X_t + W_{ho} * H_{t-1} + b_o) \\H_t &= o_t \circ \tanh(C_t)\end{aligned}$$

4.1.3. CNN-LSTM

An alternative architecture variation on the traditional LSTM that can also leverage the spatial component of the data is a CNN-LSTM. In a CNN-LSTM a CNN layer, comprised of convolution and pooling, is used to extract spatial features from the data [Sainath et al., 2015; Zhou et al., 2015]. Then those outputs from the CNN layer are fed as inputs to an LSTM, which learns the temporal relationship.

4.2. Data Processing

The input data were comprised of nine channels including the five case parameters (permeability, porosity, conductivity, bottom hole pressure, aperture size), near field pressure, far field pressure, time, and either cumulative production (for the pressure distribution model) or pressure distribution (for the cumulative production model). The pressure distribution, cumulative production, near and far field pressure, and time were dynamic inputs, that is, they had different values at different points in time. The case parameters were static inputs. Of the 32 cases in the data, three were held out as a test set. All data were scaled to be between zero and one.

4.3. Results

4.3.1. ConvLSTM

We used separate ConvLSTM networks to model the cumulative production and pressure distribution. A simple model comprised of one ConvLSTM layer, batch normalization, and two dense layers was sufficient to model both cumulative production and the pressure distribution (Figure 4-1). The cumulative production model also had a flatten layer following batch normalization. For both models the filter size was 3x3, the loss was Mean Squared Error (MSE), and the optimizer was Adam. Using the SHERPA package in Python [Hertel et al., 2020], we conducted a random search hyperparameter optimization to select the number of convolutional filters, the number of nodes in the first dense layer, and the activation functions for both dense layers. The maximum number of trials was 150, and for each trial the model was trained for 100 epochs. Optimal hyperparameter values for each ConvLSTM model are given in Table 4-1. Using the optimal hyperparameter values the models were trained for a maximum of 1000 epochs and a batch size of ten, with early stopping after a patience of 200 epochs.

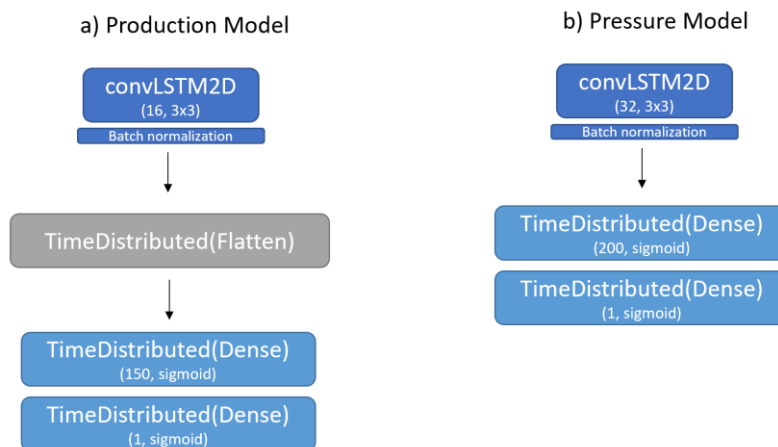


Figure 4-1. (a) ConvLSTM model for cumulative production and (b) ConvLSTM model for pressure distribution

Table 4-1. ConvLSTM hyperparameters

Hyperparameter	Search Space	Cumulative Production Optimal Value	Pressure Distribution Optimal Value
ConvLSTM filters	16, 32, 64	16	32
Dense1 nodes	50, 100, 150, 200	150	200
Dense1 activation	relu, tanh, sigmoid, linear	sigmoid	sigmoid
Dense2 activation	relu, tanh, sigmoid, linear	sigmoid	sigmoid

The actual cumulative production values and those predicted by the model for the three test cases are shown in Figure 4-2. The test MSE was 6.4×10^{-5} .

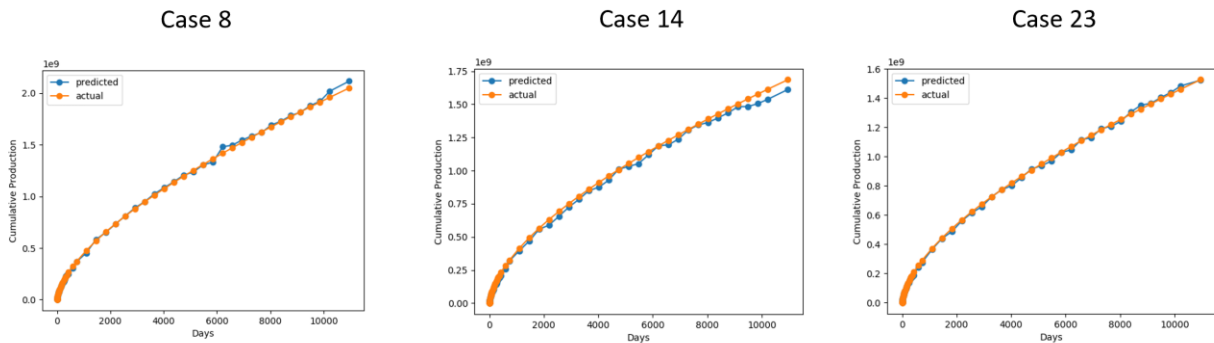


Figure 4-2. Actual and ConvLSTM predicted values of cumulative production for three test cases.

The actual pressure distribution values, those predicted by the model, and the residuals for the three test cases at three different timepoints are shown in Figure 4-3. The test MSE was 1.0×10^{-4} .

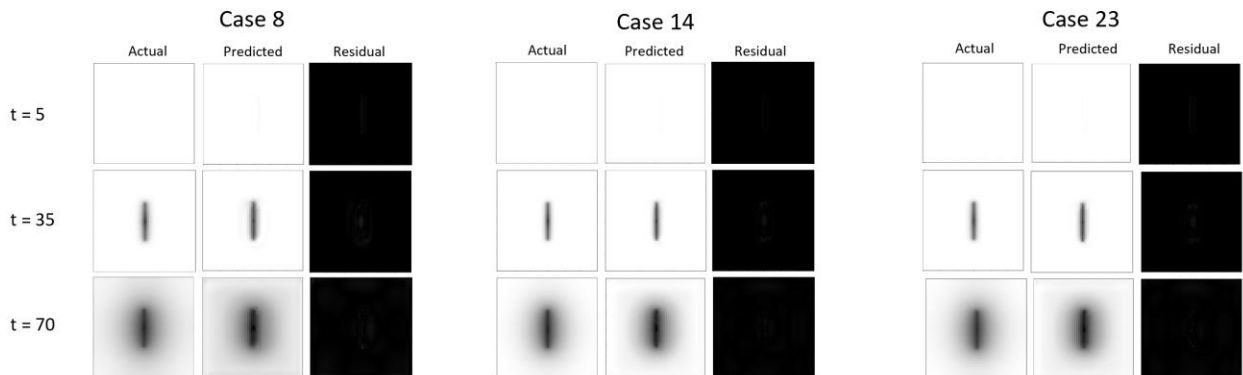


Figure 4-3. Actual and ConvLSTM predicted values of the pressure distribution for three test cases at three timepoints.

4.3.2. CNN-LSTM

We used separate CNN-LSTM networks to model the cumulative production and pressure distribution. For the cumulative production model, we used a simple CNN model comprised of one convolutional layer and pooling (Figure 4-4). For the pressure distribution model, we used a CNN with three pairs of convolutional and pooling layers. Both models also had a flatten layer before a single LSTM layer. The cumulative production model had two dense layers, and the pressure distribution had one dense layer and a reshape layer to restore the two-dimensional aspect of the predictions. For both models the convolutional filter size was 3x3, the loss was Mean Squared Error (MSE), and the optimizer was Adam.

Using the SHERPA package in Python, we conducted a random search hyperparameter optimization to select the number of convolutional filters, the number of LSTM units, the number of nodes in the first dense layer for the cumulative production model, and the activation functions for one or both dense layers for the pressure distribution and cumulative production models, respectively. The maximum number of trials was 500 for the cumulative production model and 50 for the pressure distribution model, and for each trial the model was trained for 100 epochs. Optimal hyperparameter values for each CNN-LSTM model are given in Table 4-2. Using the optimal hyperparameter values the cumulative production model was trained for a maximum of 1000 epochs and a batch size of ten, with early stopping after a patience of 200 epochs, whereas the pressure distribution model was trained for a maximum of 500 epochs with early stopping.

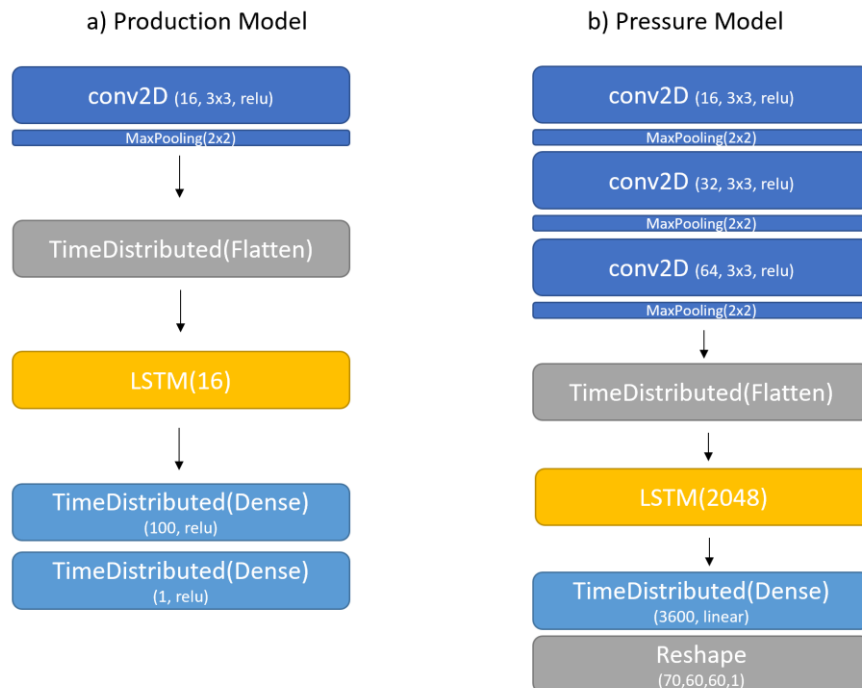


Figure 4-4. (a) CNN-LSTM model for cumulative production and (b) CNN-LSTM model for pressure distribution.

Table 4-2. CNN-LSTM hyperparameters

Hyperparameter	Search Space		Cumulative Production Optimal Value	Pressure Distribution Optimal Value
	Production	Pressure		
Convolutional filters	16, 32, 64	[16,32,64] ; [32,64,128]	16	[16,32,64]
LSTM units	16, 32, 64	512, 1024, 2048	16	2048
Dense1 nodes	50, 100, 150, 200	N/A	100	N/A
Dense1 activation	relu, tanh, sigmoid, linear	relu, tanh, sigmoid, linear	relu	linear
Dense2 activation	relu, tanh, sigmoid, linear	N/A	relu	N/A

The actual cumulative production values and those predicted by the model for the three test cases are shown in Figure 4-5. The test MSE was 5.0×10^{-4} .

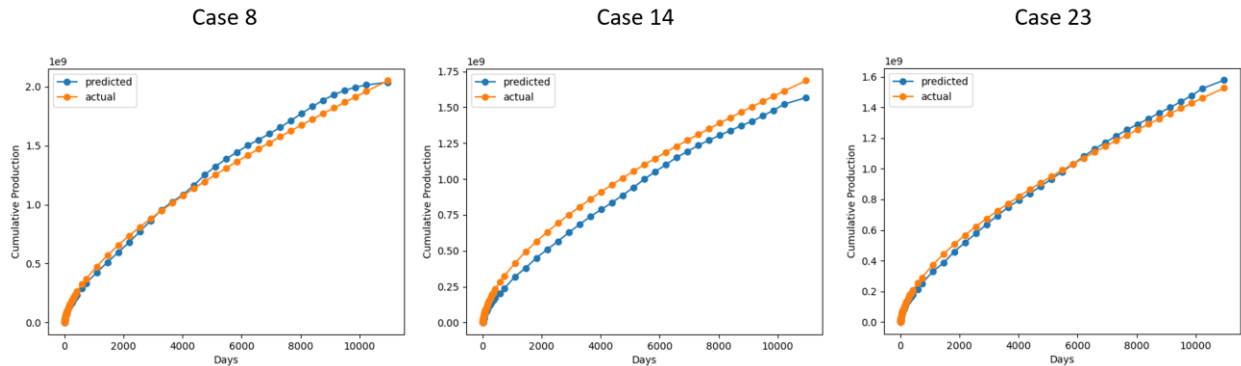


Figure 4-5. Actual and CNN-LSTM predicted values of cumulative production for three test cases.

The actual pressure distribution values, those predicted by the model, and the residuals for the three test cases at three different timepoints are shown in Figure 4-6. The test MSE was 3.0×10^{-5} .

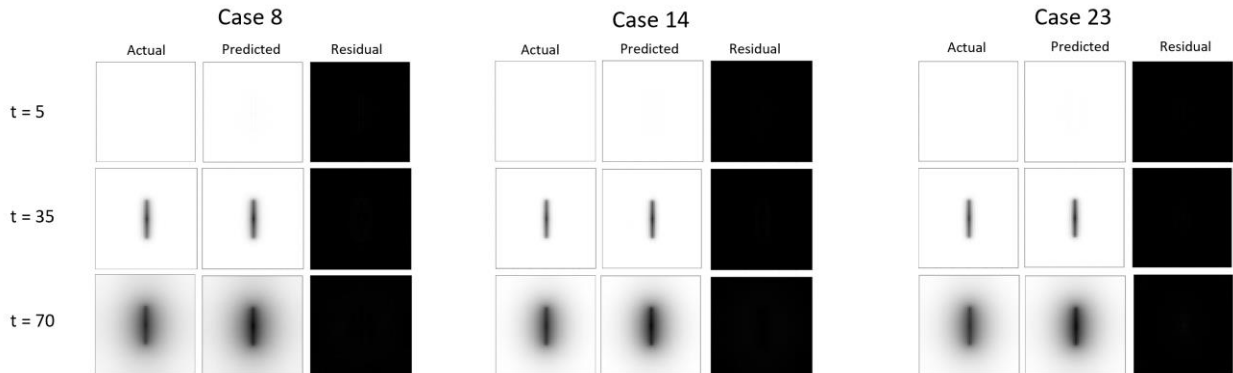


Figure 4-6. Actual and CNN-LSTM predicted values of the pressure distribution for three test cases at three timepoints.

5. REINFORCEMENT LEARNING OF OPERATION

Modern subsurface geosystem models are characterized by high-dimensional distributed input variables. Because of limited data available, especially during the planning stage, there is a need for new ways to explore, plan and monitor gas production from wells. In our work, we formulate a deep reinforcement learning (DRL) framework, in which a deep Q-learning network agent is trained to maximize rewards and optimize the production of gas obtained from horizontal wells. We use a surrogate model developed in Section 4 to lower the computational costs of the large-scale shale gas recovery model. Our surrogate model is a Convolutional LSTM that allows us to predict pressure and production transition states for the reinforcement learning model. The goal is to develop and implement the framework to obtain an optimal policy for maximizing gas production and minimizing cost for optimal geosystem management.

5.1. RL Introduction

Reinforcement learning (RL) involves an agent interacting with a dynamic environment through a sequence of actions, observations, and rewards [Sutton and Barto, 2018]. Beyond the set of available actions, the agent is not told what to do, but instead seeks to maximize the cumulative future reward by discovering an optimal policy through interactions. A Markov decision process (MDP) give us a way to formalize a problem into RL. In the MDP, we have a decision maker, called an agent, that interacts with the environment it's placed in. These interactions occur sequentially over time. At each timestep, the agent will get some representation of the current state of the environment. Given this representation, the agent selects an action to take. The environment then transitions into a new state, and the agent is given a reward as a consequence of the previous action. Throughout this process, it is the agent's goal to maximize the total amount of rewards that it receives from taking actions in given states. This means that the agent wants to maximize not just the immediate reward, but the cumulative rewards it receives over time. The diagram in Figure 5.1a, shows the steps of the RL process, taken according to MDP, listed in sequence with more descriptive detail below:

1. At time t , the environment is in state S_t .
2. The agent observes the current state and selects action A_t .
3. The environment transitions to state $S_{(t+1)}$ and grants the agent reward $R_{(t+1)}$.
4. This process then starts over for the next timestep, $t + 1$.

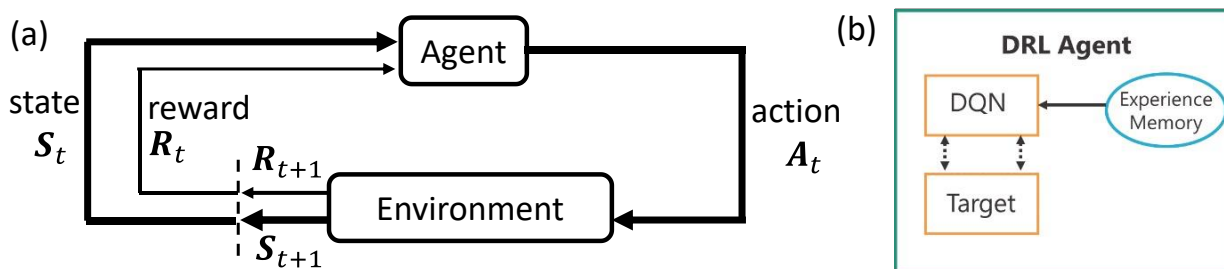


Figure 5-1. (a) The schematic of agent-environment interaction in reinforcement learning (redrawn from Sutton and Barto [2018]) and (b) the three components of the DRL agent in this work: Deep Q-Network (DQN), Target Network, and Experience Memory.

5.1.1. Q-learning

Q-learning is a form of RL. In Q-learning we have a Q-table which is a simple data structure that we use to keep track of the states, actions, and their expected rewards. More specifically, the Q-table maps a state-action pair to a Q-value (the estimated optimal future value) which the agent will learn. As the agent tries out different actions at different states through trial and error, the agent learns each state-action pair's expected reward and updates the Q-table with the new Q-value. One of the goals of the Q-Learning algorithm is to learn the Q-Value for a new environment. The Q-value is the maximum expected reward an agent can reach by taking a given action A from the state S . After an agent has learned the Q-value of each state-action pair, the agent at state S maximizes its expected reward by choosing the action A with the highest expected reward.

The Bellman Equation (5.1) tells us how to update our Q-table after each step we take. The agent updates the current perceived value with the estimated optimal future reward which assumes that the agent takes the best current known action.

$$Q(S_t, A_t) = (1 - \alpha)Q(S_t, A_t) + \alpha * \left(R_t + \lambda * \max_a Q(S_{t+1}, a) \right) \quad (\text{Eq. 5-1})$$

S = the state

A = the action taken

R = the reward from taking an action

t = the timestep

α = the Learning Rate

λ = the discount factor which causes rewards to lose their value over time so more immediate rewards are valued more highly.

As the agent learns the Q-value function, it can get stuck in local minima. To overcome this limitation, a balance between exploration and exploitation is used. A common strategy for tackling the exploration-exploitation tradeoff is the Epsilon Greedy Exploration Strategy. The exploration rate ϵ is the probability that our agent will explore the environment rather than exploit it. We want a balance between exploitation and exploration, therefore ϵ will decay by some rate that we set so that the likelihood of exploration becomes less and less probable as the agent learns more and more about the environment.

5.1.2. Deep Q-learning

In the Q-learning a table maps each state-action pair to its corresponding Q-value. In Deep Q-learning a Neural Network maps input states to (action, Q-value) pairs. The loss from the network is calculated using Equation 5.2 by comparing the output Q-values ($q(s, a)$) to the target Q-values ($q^*(s, a)$) with the objective to minimize this loss.

$$loss = q^*(s, a) - q(s, a) \quad (\text{Eq. 5-2})$$

After the loss is calculated, the weights within the network (represented as links between nodes in the DNN) are updated via Stochastic Gradient Descent and backpropagation.

5.1.3. Target network

The learning process for deep Q-learning uses two deep neural networks (DNNs). These networks have the same architecture but different weights. Every N steps, the weights from the main network

are copied to the target network. If we were to try to use a single DNN (without copying), then in order to calculate the target Q-values we would need to do a second pass through the network. This means that we would be changing the weights of our network as well. This introduces instability because as the Q-value moves closer to the target Q-value, the target Q-value moves away as well. Thus, the target network is separate from the policy network. Using both of these networks leads to more stability in the learning process and helps the algorithm to learn more effectively.

5.1.4. Experience replay

We also use Experience Replay (memory) to enhance learning progress and stability. For this enhancement, we store the agent's previous experiences at each state and randomly sample from these to train the network. This is done to break the correlation between consecutive samples (which would otherwise lead to inefficient learning). A diagram showing these three components in our RL agent is shown in Figure 5.1b.

5.1.5. Policy and policy gradients

A policy, in general, is a mapping between a certain state s and action a . It is usually represented by: $\pi(s, a)$. The goal of reinforcement learning is to optimize these policies for the maximum reward. There are several alternative formulations of policy (listed below):

- In Q-learning, the set of policies is held in a tabular form, with each state corresponding exactly with a certain action.
- In other forms of RL, the policy may be stochastic.
- The policy can also be represented in a deep neural network or some other form.

The typical way of learning a policy is to learn its value function. This function indicates how “good” it is to perform an action when in a given state s . The policy would then be derived from this information. However, the policy gradient can also be computed directly. DNN learning uses this approach through function approximation. When this approach is used, the network learns by following a gradient. A similar approach has been used for optimal monitoring system during geologic carbon sequestration [Sun, 2020]

5.1.6. Reinforce with baseline

An early policy gradient method was reinforced. It is used for episodic tasks. The method generates a complete episode, and then uses the sum of rewards to calculate the gradient. A good example for this method is Pole Cart. A baseline can be useful in RL for efficiency and stability. It was found that subtracting a baseline led to a reduction in variance and faster learning. One way to implement a baseline is to scale the returns by mean and standard deviation. In general, the baseline represents a value for comparison to the expected reward from the current choice under consideration. For example, the RL agent can compare against fully greedy play (also known as Self-Critic).

5.1.7. Trust region policy optimization (TRPO)

One issue with policy gradient learning is that sometimes the gradient leads to dramatic changes in parameters. This is not always best for learning. TRPO restricts the policy with a constraint that specifies how close new and old policies can be. It also regularizes and constrains step sizes for more regular and safe improvements.

5.1.8. Proximal policy optimization (PPO)

PPO is an easier way to implement trust regions, but it does not always perform as well as TRPO. In PPO, we limit how far we can change our policy in each iteration through comparison using the KL-divergence.

5.2. Model Description

Our goal of RL is to enhance subsurface resources recovery by optimizing well operations. Optimal well operations with proper monitoring systems will enhance the recovery while minimizing any environmental risks such as induced seismicity, integration of subsurface systems. Therefore, we want to minimize cost and maximize production of shale gas.

5.2.1. Problem setup

Our overall objective of the problem setup is to maximize a total of cumulative production of shale gas while minimizing the operation and monitoring costs. It is assumed that there is a cost associated with bottomhole injection and with the total number of wells. Initially, our reinforcement learner will not control the number of wells, and so these will be part of the environment parameterization. The RL agent will control the amount of bottomhole extraction, as its action, which can either be increased or decreased at each timestep. Our model parameters include permeability, porosity, hydraulic conductivity of fracture, bottom hole pressure, fracture aperture, and the number of horizontal wells. An example diagram of our RL agent operational framework is shown in Figure 5-2.

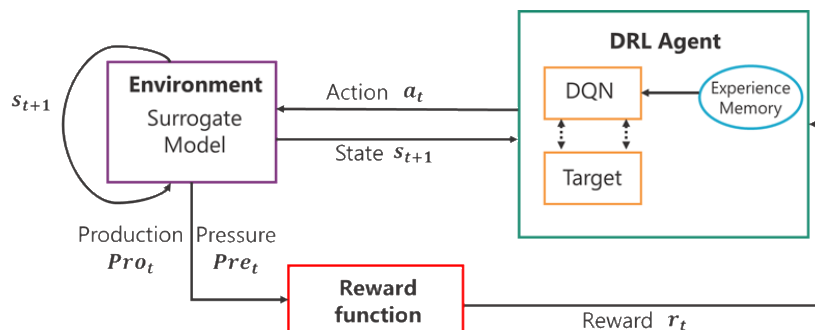


Figure 5-2. Diagram of deep reinforcement learning framework.

5.2.2. Action space

In our RL agent environment, we use the bottomhole pressure (P_{bh}) for our action space. Here we start with two simple actions: 1) add 100 to our bottomhole pressure or 2) subtract 100 from the bottomhole pressure:

$$A = \{P_{bh} + 100, P_{bh} - 100\}$$

Having a small action space may help increase the learning rate of the model. It is important to note that the reason we are using the bottomhole pressure is because of the lack of data. Optimally we would also use the number of wells as our action space, too.

5.2.3. Reward design

Our goal is to minimize the loss and maximize the benefits. This translates to determining what sequence of actions is to be taken to reach this goal. The reward function allows us to set some constraints to our model. Our reward function is shown in Equation 5.3.

$$r(\mathbf{s}_t, a_t) = \begin{cases} (price_{gas} - c_{prod})\Delta V_{gas}(\mathbf{s}_t, a_t) & \text{if } \Delta P_{well} \leq \Delta P_{max} \\ 0 & \text{if } \Delta P_{well} > \Delta P_{max} \end{cases} \quad (\text{Eq. 3})$$

$price_{gas}$: market price of natural gas per cubic meter

ΔP_{well} : the well pressure drawdown

ΔP_{max} : maximum pressure drawdown

c_{prod} : production cost of natural gas per cubic meter

$\Delta V_{gas}(\mathbf{s}_t, a_t)$: total volume of gas production

Note that zero reward is achieved when the well pressure drawdown exceeds the maximum allowable value. Well pressure drawdown is computed as the largest difference between initial and current (or final) pressure values, shown in Figure 3-1b.

5.2.4. The environment

In RL, the environment changes once you take an action. Depending on your environment it can be computationally expensive to change it. For our problem the environment consists of wells with changing pressure fields and changing production amounts. This means that we have a very complex system that has many variables. In order to lower the computability costs of running large scale system models we will use a surrogate model. The surrogate model allows us to predict pressure and production transition states. The model consists of a ConvLSTM model described in Section 4.

In the future we will use a DQN algorithm for optimizing well injection operation described in this section.

6. PHYSICS INFORMED NEURAL NETWORKS

As universal function approximators, neural networks can learn the mapping inputs to outputs. When solving regression problems using supervised learning, neural networks minimize a loss function by adjusting the weights in each layer. These weights are randomly initialized, and then adjusted using gradient descent and backpropagation during training [Rummelhart et al., 1986]. The performance of this learning process depends on the amount and quality of training data available. Sufficiently complex neural networks (in number of layers and neurons) can minimize errors during training. This can sometimes lead to over-fitting of data. To avoid this, a test set is set aside to ensure that the neural network can generalize well over this set. Regularization is another approach used to prevent over-fitting and add structure to the function approximation [Goodfellow et al., 2016]. These regularizations are added to the objective function as extra loss terms. Examples of these terms include L1 and L2 norms and dropouts. These extra loss terms have the effect of shaping the space of parameters, and hence the function approximation that is learned, by biasing it towards a certain structure.

Often, we do not have sufficient knowledge of the process that generated the data and regularization should be used carefully to not introduce bias. However, if something is known about the underlying process, then the appropriate information can be added as extra loss functions to help the learning process incorporate this structure. This latter scenario is useful when there is limited data available for training [Perdikaris and Raizi, 2019]. By leveraging the known underlying relationship between variables, fewer training data can be used to learn the function of interest using neural networks.

6.1. PINN Introduction

Physics informed neural network (PINN) uses known physics constraints and laws from physics to define relationship between predicted and input variables. These constraints bias the function approximation to capture the underlying physical patterns while mapping the inputs to output data collected from a physical process [Raizi et al., 2019]. If the full PDE is used to enforce constraints, then the model learns the underlying physics and the model can learn to predict the solution of the PDE for arbitrary inputs in the domain without needing to explicitly solve it! This model can then be used for rapid inference in several scenarios, including as an environment for reinforcement learning. PINN models have been developed and used for various applications such as managing unconventional reservoirs [Mudunuru et al., 2020] and hydrology [Wang et al., 2020]. Other approaches to PINN include uncovering the underlying PDE model from data. However, in this effort we focus on the former.

These physics informed laws are used as additional loss functions (usually with mean square error loss) during the neural network training process. Often these laws relate variables through differential operators. Thus, implementing these loss functions require differentiation of the predicted data with respect to input or other predicted variables. Recent availability of neural network libraries such as TensorFlow and PyTorch implement numerical differentiation to compute the loss component at each layer using backpropagation [Rummelhart et al., 1986]. By exposing this feature to users, neural network models can now use these functions to differentiate data tensors. Thus, to train a PINN to learn the solution for partial differential equations, additional losses are added to the standard loss from data reconstruction. These additional losses enforce the equations, and initial and boundary conditions for appropriate data points.

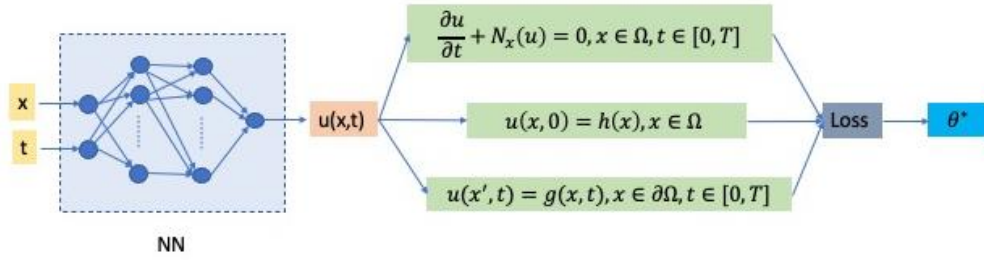


Figure 6-1. Schematic of PINN architecture (modified from Rossi et al., 2019).

Figure 6-1 shows the process of PINNs. The grey-blue NN on the left learns the mapping from \mathbf{x} , \mathbf{t} to $u(\mathbf{x}, \mathbf{t})$ by minimizing the mean square error (MSE, not shown here) and the auxiliary losses described by the PDE equations and boundary/initial conditions shown in green boxes. The top equation is the PDE and the lower two green equations represent the initial and boundary conditions. This combined loss is used to update the parameters of the NN. While implementing the above in TensorFlow 2.5, the equations in green boxes above are implemented using the GradientTape functionality [TensorFlow documentation, 2021] to calculate the partial derivatives. As an example, we train using data generated by a one-dimensional Burger’s equation [Burger, 1948] described below.

6.1.1. Burger’s equation

Burger’s equation describes a one-dimensional diffusion process in fluids and can be generalized to higher dimensions. It is used for modeling gas dynamics, fluid flow, acoustic waves, etc. It is given by the following partial differential equation:

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} - \nu \frac{\partial^2 u}{\partial x^2} = 0$$

where a diffusion coefficient $\nu \geq 0$. When the coefficient is 0, the equation describes an inviscid flow.

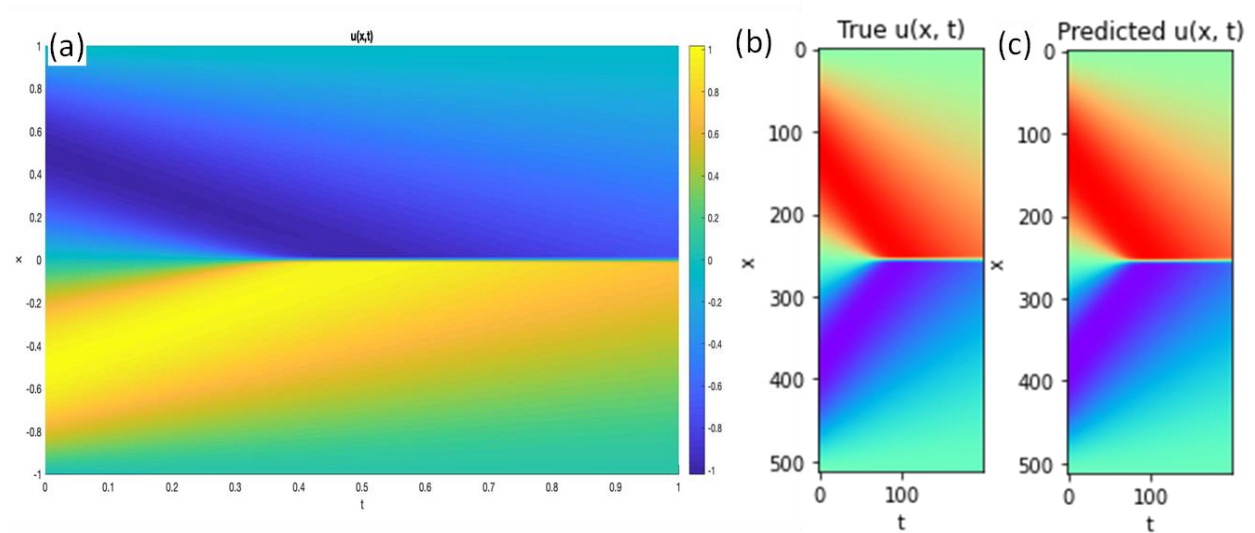


Figure 6-2. (a) An example of solution to Burger’s equation over $u(\mathbf{x}, \mathbf{t})$ and (b-c) comparison of true solution and PINN prediction. The domain size is 500 (\mathbf{x}) x 200 (\mathbf{t}) points.

Figure 6-2a shows the solution to burger’s equation. The horizontal axis shows how the amplitude evolves over time at each position point (shown along the vertical axis). The initial conditions in this case are given by a sinusoidal amplitude over x . That is $u(x, 0) = -\sin(\pi x)$ and the boundary conditions are fixed at 0 for all t . The diffusion coefficient is $\nu = \frac{0.01}{\pi}$.

Training data is generated by numerically solving the above equation with the chosen initial and boundary conditions. Using this training data, the neural network then learns to fit the data with an MSE loss and the PDE losses. This process requires minimal preprocessing. The input features are normalized to bring them to the same scale since $x \in [-1, 1]$ and $t \in [0, 1]$. The output is a one-dimensional function u that takes values in the range $[-1, 1]$. As an exemplar, a simple fully neural network with 8 hidden layers and ReLU activation function was set up and trained. The network learns the mapping as shown in Figure 6-2b. Note that the PINN result matches the true profile very well.

6.2. Stokes’ Flow Equation

As a problem relevant to subsurface energy production, we applied PINNs to Stokes flow. This non-trivial system was used to describe viscous steady state flow of incompressible fluids in porous media [e.g., Yoon et al., 2012; 2015]. Stokes flow is described the following set of vectorized equations.

$$\begin{aligned} \mu \nabla^2 \mathbf{u} - \nabla p + \mathbf{f} &= \mathbf{0} \\ \nabla \cdot \mathbf{u} &= 0 \end{aligned}$$

To build and train a PINN, we first generated data for the domain shown in figure 6-3a. This 2D domain represented a flow in a rectangular section with a cylinder obstacle. The fluid flow is from left to right as shown by the inlet and outlet. Here, $\mathbf{u}(\mathbf{x}, \mathbf{y})$ is a vector of velocities in the x and y direction ($\mathbf{u}_x, \mathbf{u}_y$).

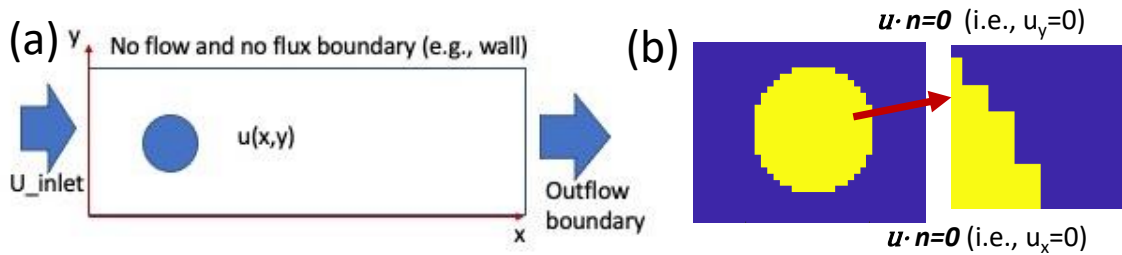


Figure 6-3. (a) Numerical domain, single cylinder obstacle, and flow direction of the stokes flow simulation, and (b) staircase discretization of the cylinder obstacle with zero velocity boundary conditions normal to the face of boundary.

To generate training data, the rectangle is chosen to 0.04 m long x 0.01 m high. U_{inlet} is fixed at 0.001 m/s. The domain is discretized with a mesh size of $dx = dy = 0.0002$ m, resulting in 200 x 50 cells. Dynamic viscosity (μ) = 0.001 Pa·s is used. The discretization results in the pixelation of the cylinder boundary as in Figure 6-3b. The following boundary conditions are enforced (inlet and outlets, no flux boundary, no slip boundary at walls and cylinder obstacle):

$$u_x(x = 0, y) = u_{inlet}, \frac{du_x}{dx} = 0 \text{ at } x = L$$

$$\frac{du_y}{dx} = 0 \text{ at } y = 0 \text{ and } y = H$$

$$u_x = 0 \text{ at } y = 0 \text{ and } y = H$$

$$u \cdot n = 0, \text{ i. e., } u_x = u_y = 0 \text{ around the obstacle}$$

Since it is a steady state flow, there are no initial conditions and no explicit dependence on time. Using the above setup, the PDE is numerically solved to generate the training data. The results of the velocity data along with the flow lines for ease of visualization is shown below.

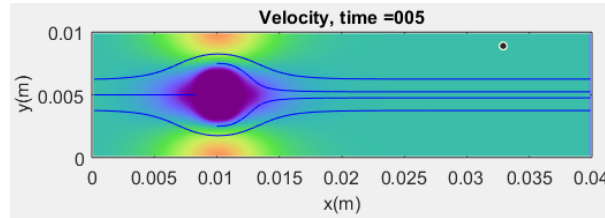


Figure 6-4. Stokes flow simulation in the presence of single cylinder obstacle. Solid line represents streamlines.

6.2.1. PINN training

Despite the lack of explicit time dependence, stokes flow model is more complicated than the Burger's equation from section 6.1. Stokes flow deals with problems in higher dimensions and describes changes in pressure and velocity (along x and y components). The relationships among the variables are captured in two equations. There are also boundary conditions along the top and bottom walls, inlets, outlets, and the boundary along the cylinder.

To learn this model, the neural network takes in two inputs, viz., the x and y co-ordinate information and predicts 3 outputs: x and y components of the velocity vector and the pressure at each location. It is important to note that unlike Burger's equation, the magnitudes of the predicted variables are at different scales. To avoid imbalanced contributions to the loss components, the values are normalized using min-max scaling.

Training is achieved by the following process:

1. First the training data is generated using numerical simulation of the PDE
2. The data is normalized using min-max scaling, i.e., $x_s = \frac{x - x_{min}}{x_{max} - x_{min}}$
3. The data is then sampled from the domain and the data under the cylinder is discarded
4. Since the boundary loss is only accounted for along the boundary data points, the values at the boundaries (walls and cylinder) need to be over-sampled.
5. A sufficiently complex neural network is then set up— a fully connected neural network with 2 inputs, 3 outputs, and 8 hidden layers with 20 ReLU neurons each worked well for the problem above.
6. The various losses are coded in. The boundary losses only apply to the boundary points and need to be carefully checked. Since the data is normalized, it needs to be rescaled to the original

scale before applying the PDE loss. Alternatively, a rescaled version of the PDE loss can be used.

7. The neural network is trained for a sufficient number of epochs as determined by inspecting the losses.

After training the model learns to predict all three variables accurately as shown in Figure 6.5:

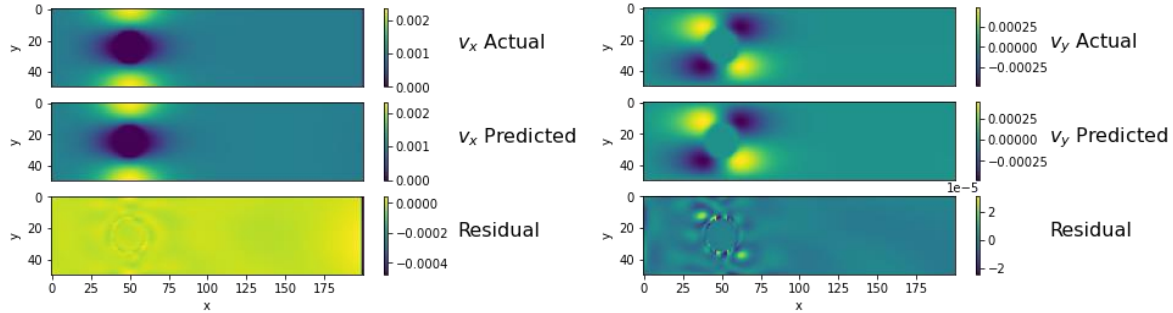


Figure 6-5. Actual and ML predicted velocities with residual difference along the x and y directions.

The results of the predicted velocities were rescaled to the original scale for comparison.

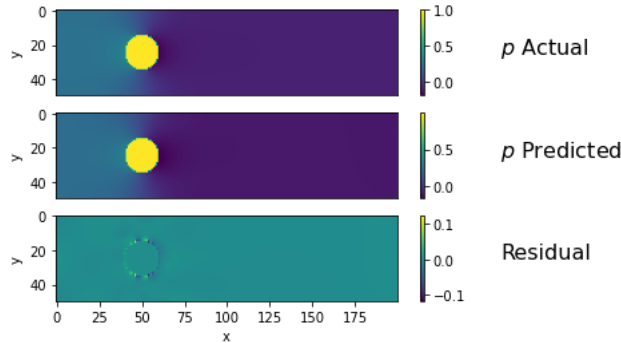


Figure 6-6. Actual and ML predicted pressure field.

The above results demonstrate the ability of neural networks as function approximators to learn solutions to PDEs accurately. Care should be taken to properly preprocess the data and the losses in terms of scaling. The individual loss components may have to be appropriately weighted to ensure that the learning process can incorporate all loss components.

When arbitrary boundaries and obstacle shapes and locations are used, determining the data points along the boundary is hard to determine and may require manual process of the data. This further increases the processing time during learning if a data point must be classified as being a boundary point or not.

As a follow-on effort, we need to explore how to generalize the solution of these PINNs to arbitrary locations and numbers of obstacles. This will require a large number of samples, an ability to encode obstacle size and location information, and the ability to automatically determine boundary locations. If successful, such PINNs can be used to quickly determine solutions to various conditions without having to solve computationally intensive numerical algorithms.

7. CONCLUSIONS

This project provided novel applications of machine learning/deep learning (ML/DL) methods for multiscale rock images, synthetic reservoir simulation cases, optimal well operation, and governing equations of flow dynamics. It is currently difficult to process multiscale images for material property estimation (e.g., image processing, permeability estimation) to develop accurate and fast reduced order models, and to account for physical constraints (e.g., governing equations, boundary conditions, known physical properties) in ML/DL models. We established a framework to develop and apply various DNN approaches for various tasks that needed to be accounted for to develop an integrated production prediction platform. During this 3-year project our technical accomplishments can be listed as:

1. We comprehensively evaluated well-known DL methods including four different 2D CNN architectures as well as a 3D CNN architecture (U-Net) using four rock image datasets including sandstone, carbonate chinks, and shale. Our results indicate that all DL models evaluated in this study outperformed manual segmentation and more complex DL models tend to outperform simple models. Transfer learning with pre-trained weights improved model performance with improved training efficiency. The ensemble approach from single trained model in contrast to commonly used ensemble approach with multiple models shows the significant improvement of model performance compared to single trained best model. More interestingly, an ensemble solution of simple U-Net performed very well even comparable with more complex U-Resnet and MultiResUnet. This promising result clearly demonstrates that the use of ensemble approach from single training process can be applicable for many other classification and regression problems [Yoon and Ringer, in prep].
2. Physics-guided machine learning is a promising approach to improve model accuracy and stable training process. We demonstrated this principle by incorporating physical properties such as drainage connectivity information for DCGANs into network generation with very high frequency features (nodes, connection edges, and complex connectivity) [Kim et al., 2021a] and measurable physical properties such as porosity and surface area into permeability prediction based on porous media images using CNN models [Yoon et al., 2020; 2021]. In addition, SAGANs was developed to improve model generation capabilities to extend the scale of image while maintaining statistical properties [Kim et al., 2021b].
3. We started evaluating multiple ML/DL methods with a simple problem where shale production data was generated using a shale reservoir simulator. The methods evaluated include artificial neural network, ConvLSTM, and CNN-LSTM. Both ConvLSTM and CNN-LSTM predicted pressure distribution and production rate over time excellently. In particular, optimal hyperparameters were obtained via automatic hyperparameter optimization process. This demonstrates the optimal training of various ML/DL models to develop reduced order models that can be applicable as a fast proxy model for more realistic unconventional resources recovery. We also generated training data from coupled Multiphysics simulations including multiscale flow in shale reservoirs with geomechanical responses during shale resources recovery processes. This dataset will be utilized to demonstrate how ML/DL models can be applied for Multiphysics problems in the future.
4. We explored the case studies to evaluate the capability of PINNs and RL approaches. These two emerging approaches against toy problems can be applicable to improve scientific machine learning and optimization for complex systems, respectively.

Finally, the outcome of this project can lead us to improve scientific machine learning approaches for complex natural and environmental systems related to energy and national security problems. Our novel approaches including physics-guided machine learning methods (e.g., connectivity-guided drainage network generation, permeability estimation with measurable quantities) and ensemble approach will be applicable for more broad classification and regression problems.

REFERENCES

- [1] Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2): 157-166.
- [2] Burgers, J. M. (1948). A mathematical model illustrating the theory of turbulence. In *Advances in applied mechanics* (Vol. 1, pp. 171-199). Elsevier
- [3] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition* (pp. 248–255).
- [4] DOE ASCR (2018), Scientific Machine Learning Pre-Workshop Report: Factual Status Document, Advanced Scientific Computing Research (ASCR) Applied Math Scientific Machine Learning Study Group, January 23, 2018
- [5] Goodfellow, I, Bengio, Y., and Courville, A. (2016). “Deep Learning”. The MIT Press
- [6] Gostick, J., Aghighi, M., Hinebaugh, J., Tranter, T., Hoeh, M. A., Day, H., ... & Lehnert, W. (2016). OpenPNM: a pore network modeling package. *Computing in Science & Engineering*, 18(4), 60-74.
- [7] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).
- [8] Hertel, L., Collado, J., Sadowski, P., Ott, J., and Baldi, P. (2020). Sherpa: Robust hyperparameter optimization for machine learning. *SoftwareX*, 12, p.100591.
- [9] Hochreiter, S., and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8): 1735-1780.
- [10] Kadeethum, T., Ballarin, F., Choi, Y., O'Malley, D., Yoon, H., and Bouklas, N. Non-intrusive reduced order modeling of natural convection in porous media using convolutional autoencoders: comparison with linear subspace techniques (under review, *Advanced in Water Resources*).
- [11] Kim, S.E., Seo, Y., Hwang, J., Yoon, H., and Lee, J. (2021a). Connectivity-informed drainage network generation using deep convolution generative adversarial networks. *Scientific Reports* 11, 1519. <https://doi.org/10.1038/s41598-020-80300-6>.
- [12] Kim, S.E., Yoon, H., and Lee, J. (2021b). Fast and Scalable Earth Texture Synthesis using Spatially Assembled Generative Adversarial Neural Networks, *Journal of Contaminant Hydrology*, Volume 243, 103867, <https://doi.org/10.1016/j.jconhyd.2021.103867>.
- [13] LeCun, Y., Bengio, Y. and Hinton, G. (2015). Deep learning. *Nature*, 521(7553), pp.436-444.
- [14] Lie, K.-A., Krogstad, S., Ligaarden, I. S., Natvig, J. R., Nilsen, H. M., and Skaflestad, B. (2012). Open source MATLAB implementation of consistent discretisations on complex grids. *Comput. Geosci.*, Vol. 16, No. 2, pp. 297-322, 2012. DOI: 10.1007/s10596-011-9244-4.
- [15] Ling, J., Kurzawski, A., and Templeton, J. (2016). Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, 807, 155-166.
- [16] Mudunuru, M. K., et al. (2020) “Physics-informed Machine Learning for Real-time Unconventional Reservoir Management”. In *proceedings of the AAAI 2020 Spring Symposium on Combining Artificial Intelligence and Machine Learning with Physics Sciences*. LA-UR-19-31611. Los Alamos National Laboratory, NM, USA. http://ceur-ws.org/Vol-2587/article_10.pdf.

- [17] Park, S. W., Lee, J., Yoon, H., and Shin, S. (2021). Microfluidic Investigation of Salinity-Induced Oil Recovery in Porous Media during Chemical Flooding. *Energy Fuels*, 35, 6, 4885–4892. <https://doi.org/10.1021/acs.energyfuels.0c04320>.
- [18] Perdikaris, P., and Raizzi, M. (2019). “SC15-009: Recent Advances in Physics-Informed Deep Learning”. 15th U.S. National Congress on Computational Physics. Lecture Notes.
- [19] Raizzi, M., Perdikaris, P., and Karniadakis, G.E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*. 378: 686-707.
- [20] Ronneberger, O., Fischer, P., and Brox. T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. p. 234–241. Springer, Cham.
- [21] Rumelhart, D.E., Hinton, G.E., and Williams, R.J. (1986). Learning representations by back-propagating errors. *Nature*. 323, 533-536. doi: 10.1038/323533a0.
- [22] Sainath, T., Vinyals, O., Senior A., and Sak, H. (2015). Convolutional, long short-term memory, fully connected deep neural networks. In the proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing; South Brisbane, Australia. 19–24 April 2015.
- [23] Salama, A., Amin, M. F. E., Kumar, K., and Sun, S. (2017). Flow and transport in tight and shale formations: A review. *Geofluids*, <https://doi.org/10.1155/2017/4251209>.
- [24] Shi, X., Chen, Z., Wang, H., Yeung, D., Wong, W., and Woo, W. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *Proceedings of the 28th International Conference on Neural Information Processing Systems*, 1, 802-810.
- [25] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
- [26] Sun, A.Y. (2020). Optimal carbon storage reservoir management through deep reinforcement learning. *Applied Energy*, 278, p.115660.
- [27] Sun, A. Y., Yoon, H., Shih, C.-Y., and Zhong, Z. (2021) “Applications of physics-informed scientific machine learning in subsurface science: A survey.” in A. Karpatne, R. Kanan, and V. Kuma, eds., *Science-guided Machine Learning: Emerging Trends in Combining Scientific Knowledge with Data-driven Methods*: CRC Press. [In Press]
- [28] Sutton, R. S., and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.
- [29] TensorFlow documentation, <https://www.tensorflow.org/guide/autodiff>. Accessed on September 06, 2021.
- [30] Wang, B. (2020). MRST-Shale: An Open-Source Framework for Generic Numerical Modeling of Unconventional Shale and Tight Gas Reservoirs. Preprints, 2020010080 (doi: 10.20944/preprints202001.0080.v1).
- [31] Wang, N., Zhang, D., Chang, H., and Li, H. (2020). Deep learning of subsurface flow via theory-guided neural network. *Journal of Hydrology*. 584. doi: 10.1016/j.jhydrol.2020.124700.
- [32] Wu, J., Yin, X., and Xiao, H. (2018). Seeing permeability from images: fast prediction with convolutional neural networks. *Science bulletin* 63(18): 1215-1222.
- [33] Yang, R., Huang, Z., Yu, W., Li, G., Ren, W., Zuo, L., ... and Sheng, M. (2016). A comprehensive model for real gas transport in shale formations with complex non-planar fracture networks. *Scientific reports*, 6, 36673.

- [34] Yoon, H., Valocchi, A.J., Werth, C.J., and Dewers, T. (2012). Pore-scale simulation of mixing-induced calcium carbonate precipitation and dissolution in a microfluidic pore network. *Water Resources Research*, 48(2). W02524, doi:10.1029/2011WR011192.
- [35] Yoon, H., and Dewers, T.A. (2013). Nanopore structures, statistically representative elementary volumes, and transport properties of chalk. *Geophysical Research Letters*, 40(16), pp.4294-4298.
- [36] Yoon, H., Kang, Q., and Valocchi, A.J. (2015). Lattice Boltzmann-based approaches for pore-scale reactive transport. *Reviews in Mineralogy and Geochemistry*, 80(1), pp.393-431.
- [37] Yoon, H., Melander, D.J., and Verzi, S.J. (2020). Permeability Prediction of Porous Media using Convolutional Neural Networks with Physical Properties. Proceedings of the AAAI 2020 Spring Symposium on Combining Artificial Intelligence and Machine Learning with Physics Sciences, (No. SAND2020-3557C). Sandia National Lab. (SNL-NM), Albuquerque, NM (United States).
- [38] Yoon, H., Melander, D.J., and Verzi, S.J. (2021). Machine Learning Application for Permeability Estimation of Three-Dimensional Rock Images. Proceedings of the AAAI 2021 Spring Symposium on Combining Artificial Intelligence and Machine Learning with Physical Sciences, (No. SAND2021-3404 C). Sandia National Lab. (SNL-NM), Albuquerque, NM (United States).
- [39] Yoon, H., and Ringer, J. R., Ensemble and multiscale image segmentation using deep learning methods, in prep.
- [40] Zhou, C., Sun, C., Liu, Z. et al. (2015). A C-LSTM neural network for text classification. *Computer Science*, 1, 39-44.

DISTRIBUTION

Email—Internal

Name	Org.	Sandia Email Address
Hongkyu Yoon	08864	hyoon@sandia.gov
Austin A Holland	08864	aaholla@sandia.gov
Steve Kleban	08722	sdkleba@sandia.gov
John S Wagner	01421	jswagne@sandia.gov
Benjamin K Cook	08900	bkcook@sandia.gov
Susan J Altman	08140	sjaltma@sandia.gov
Technical Library	01977	sanddocs@sandia.gov

This page left blank

This page left blank



**Sandia
National
Laboratories**

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.