Physics Informed Deep Learning for Flow and Transport in Porous Media

Cedric Fraces Gasmi
cfraces@stanford.edu

Jihoon Park
jhpark3@stanford.edu

Dong Hee Song
dhsong@stanford.edu

Abstract

We propose a new method for the forecasting of flow and transport problem in porous media. The method is a fast and efficient way to generate surrogate responses to the various realizations of a simulation model. It is an alternative to the CPU and time consuming finite volume simulation of the pressure and fluids concentrations. The model is built using a convolutional network encoder decoder architecture that takes the reservoir permeability as an input and outputs the pressure and saturation fields at various time steps. We propose to solve the SPE10 reservoir simulation case (top layer) and various delineations of it (1000 realizations). We cast the problem as an image to image regression. We predict the distribution of two quantities of interest (pressure and saturation) after training on a few timesteps for a few hundred realizations and then predict on unseen realizations and time steps. The result is a neural network based simulator that honors the very nonlinear nature of the flow with a high fidelity and can be run in a fraction of the time required for the classical finite volume scheme. This type of high fidelity surrogate can be used for uncertainty quantification which typically requires a large number of function evaluations.

1. Introduction

Data driven models are widely used for reservoir forecasting problems and uncertainty quantification. Such models are important as they support decision making in the face of risk. A key problem in subsurface flow is to resolve concentration shocks and rarefaction waves resulting from the multi-phase transport conservation laws (so-called Buckley-Leverett solution) to the Riemann problem. Classical surrogate models (Gaussian processes, polynomial chaos expansion) are commonly used but fail to capture non linear behavior for large dimensional systems. This project proposes a new hybrid machine-learning/physics-based approach to reservoir modeling. The model is a neural network that is jointly trained to match any available experimental data and respect governing physical laws. The input of this method are images of permeability fields. The network architecture can be easily modified to adjust to different image sizes. We then use a network based on encoder-decoder architecture with a modified loss function to predict the space and time evolution of pressure and saturation within the reservoir. The modification to the loss function encourages physical solutions by penalizing deviations from the governing physics. The training data was generated by using state of the art simulation methods.The proposed methodology is a simple and elegant way to instill physical knowledge to machine-learning algorithms. This method aims to alleviate two significant shortcomings of machine-learning algorithms: the requirement for large datasets and the reliability of extrapolation. We essentially draw from the work of Perdikaris *et al*. [14] and Zabaras *et al*. [10]. The input to our algorithm is a permeability field that is intepreted as an image with only one color channel. We use a CNN encoder-decoder architecture to output a predicted pressure and saturation field at various time steps. The trained model is able to match the simulator results to a high degree of accuracy. The hyper-parameters were tuned with experiments. The principles presented in this paper can be applied in innumerable ways in the future and should lead to a new class of algorithms to solve both forward and inverse physical problems.

2. Related Work and Background

Data driven approaches have been supporting decisions in the oil and gas industry since its inception. Arps introduced decline curve analysis (DCA) in 1945 [1]. DCA is based on empirical observations that the decline rate of production in oil reservoirs follow exponential, harmonic, or hyperbolic decline rates. Later, the exponential decline was found to be physically based in the production of a single phase fluid from a closed reservoir for through the works of Brons(1963)[2], Fetkovich *et al*. (1971)[4], and Fetkovich *et al*. (1980)[5]. Furthermore, justification of deviation from the exponential decline model is provided in chapter 9 of Lake(2009)[8]. Models based on physics and data have been at the driving the oil and gas industry.

The current industry standard for models is reservoir simulation. A common workflow is to build a simulator based on a geological realization (geomodel) and then up-

date this simulator based on actual production data. These simulators are typically based on the finite volume approach to solving PDE's and have gained massive popularity as computation power became more accessible. The strength of simulators is that multiple physical effects and non-linearities can be computed [9]. However, this process can be computationally expensive, and this cost is amplified when many runs are required of the simulator (e.g. uncertainty quantification based on Monte Carlo methods). The computational complexity of full numerical simulation typically grows quadratically with the number of degrees of freedom N as the linearization of the system of PDE's requires the computation of a jacobian matrix that has N^2 elements.

Advancements in deep learning methods have inspired attempts to replace some of the tasks traditionally performed by simulation with neural network based methods. The works of Perdikaris *et al*. [14, 15] and Rassi[13] have introduced physics informed neural networks (networks that learn while adhering to physics) and networks which honor the governing physics itself. Zabaras *et al*. [10] uses a encoder-decoder network architecture to create surrogate models for predicting the pressure and saturation evolution in time and space. In another work, Zabaras also introduces a methodology which incorporates the governing equations of the physical model in the loss functions [18]. Karpatne *et al*. [7] have introduced another framework for combining physical knowledge with neural networks using a physics-based loss function. This study draws inspiration from the work done by Perdikaris and Zabaras.

3. Methods

The goal is to construct a surrogate model of reservoir simulator by performing image-to-image regression with CNN. We follow the formulation proposed by [10]. Input $x \in \mathbb{R}^D$ is log-permeability field (unit of log miliDarcy, lnmd), where $D = H \times W = \text{height} \times \text{width}$. The outputs consist of three variables which are spatial-temporal pressure map $P^{i,j} \in \mathbb{R}^{D \times T}$, saturation map $S^{i,j} \in \mathbb{R}^{D \times T}$, and another spatial-temporal variable $\xi^{i,j} \in \mathbb{R}^{D \times T}$, which is binary variable that indicates whether water invades a certain cell. Here, the subscripts i and j are used to denote pixel and time, respectively. It should be noted that each input and output are considered as time-dependent images. As a result, the deep neural network model f can be formulated as:

$$\hat{y}^{i,j} = \mathbf{f}(\mathbf{x}^i, t_j; \theta) \tag{1}$$

Where θ: neural network parameters, t_j: specified timestep.

The variable $\xi^{i,j}$ is defined as:

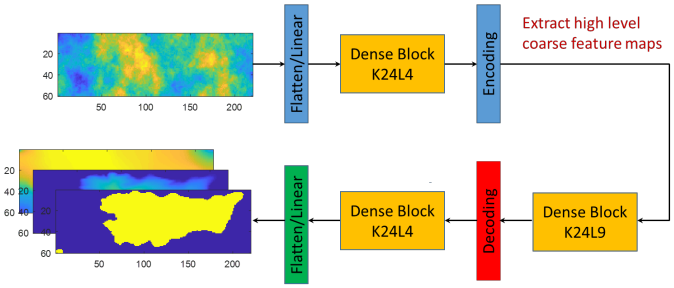

Figure 1. Network configuration used for training. It is modified from [10]. L and K means the number of internal layers and constant growth rate, respectively

$$\xi^{i,j} = \begin{cases} 1 & S^{i,j} > 0.2 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

As seen in Eq. (2), ξ depends on $S^{i,j}$. This variable is introduced by [10] to describe discontinuous waterfront. The training is performed in two stages, the first stage to learn Eq. (1) and the second stage to learn Eq. (2). The objective functions at each stage are mean squared error and binary cross entropy, see Section 5 for details. We use batch normalization and Adam optimizer to minimize both objective functions.

Every pixel value of the permeability field map be perceived as one pixel of an image (single channel). In this research, we utilize the neural network design proposed by [10]. However, their images have dimensionality of 50 x 50, whereas the dimensionality of reservoir differs by cases. In our example, which is SPE10 field, the dimensionality is 60 x 220. One of contributions of this research is to explore the possibilities of transfer learning instead of designing the neural network configurations from scratch. By doing so, the learning can be easily generalized.

The network architecture is based on deep convolutional encoder-decoder network [10]. Figure 1 illustrates the network architecture trained in this report. The first flatten and dense layer (linear) is to decrease the dimensionality of inputs from 60 x 220 to 50 x 50. Another way to do achieve this is to redesign filter and strides for convolutional layer. However, we do not take this approach because the preexisiting network was already optimized for square images and filters.

A dense block consists of consecutive layers performing batch normalization, ReLU and convolution. The outputs of dense block are fed to encoding or decoding layers. Consecutive dense blocks and encoding layers are to extract high level features from permeability maps with dimensionality reduction. Repeated decoding layers are to map the low dimensional features to outputs. Table 1 summarizes configuration of neural network architecture.

Table 1. Network architecture used in this report. This is modified from [10]. N_{out} denotes the number of output feature map. k'= kernel size, s =stride, p= padding.

Layers	N_{out}	Output dimension
Input	1	60×220
Flatten Layer	1	13,200
Dense Layer	1	2,500
Reshape Layer	1	50×50
Convolution ($k'7s2p3$)	48	25×25
Dense Block 1($K24L4$)	144	25×25
Encoding Layer	72	13×13
Dense Block 1($K24L9$)	289	13×13
Decoding Layer 1	144	25×25
Dense Block 3 ($K24L4$)	240	25×25
Decoding Layer 2	3	50×50
Flatten Layer	1	7,500
Dense Layer	1	39,600
Reshape Layer	3	60×220
Normalization Layer	3	60×220

4. Dataset and Features

In this report, we use the top layer of SPE10 field [17]. SPE10 field has the dimensionality of 60 x 220 and it is assumed that log permeability follow multivariate normal distributions, see Figure 2 (left). 1,000 realizations of permeability field are generated by Monte Carlo sampling and output pressure and saturation maps are obtained by solving the governing PDE's and relationships for reservoir simulation provided in Equation 3. The PDE can be solved by running the commercial reservoir simulator, Schlumberger ECLIPSE [16].

$$\begin{aligned} \boldsymbol{\nabla} \cdot [\lambda_w(\boldsymbol{\nabla} \cdot p_w - \gamma_w \boldsymbol{\nabla} \cdot z)] &= \frac{\partial}{\partial t}\left[\phi \frac{S_w}{B_w}\right] + q_w \\ \boldsymbol{\nabla} \cdot [\lambda_o(\boldsymbol{\nabla} \cdot p_o - \gamma_o \boldsymbol{\nabla} \cdot z)] &= \frac{\partial}{\partial t}\left[\phi \frac{S_o}{B_o}\right] + q_o \\ P_c = p_o - p_w &= f(S_w) \\ S_w + S_o &= 1 \end{aligned} \tag{3}$$

Where, λ: mobility, S: saturation, γ: specific gravity, B: formation volume factor, q: production rate, P: pressure, P_c: capillary pressure. Subscripts w and o indicate water and oil, respectively. Mobility λ is proportional to permeability k. We are interested in mapping k to P and S.

The water is injected at the center and fluids (oil and water) are produced at four corners of the rectangular reservoir. However, due to spatial variation of permeability, the water propagation is asymmetric. Water tends to propagate to high permeability region, see Figure 2 for such examples. The time for training is chosen as 50, 100, 200, 500, 1,000 and 2,000 day, see Figures 7 and 8 which show how pressure, saturation, and waterfront evolve over time. 1,000 samples are split to various ratios (800/200, 700/300,..., 200/800) to prepare training and test sets.

5. Experiments/Results/Discussion

For this analysis, we focus on the 2D results with a mapping of pressure and saturation fields through time. The case we present is more complex than the one presented in [10] and we had to add extra-layers to the existing architecture (see Method section). Because of the non-linear nature of the PDE's governing this flow problem, slight variations in the input permeability field lead to very different outcome in terms of pressure and saturation maps as seen in Figure 2. Some plumes percolate to all 4 wells while others are routed to one or two wells. This is due to the dependency of relative permeability on saturation. We train our model on a fixed set of realizations at given time steps and predict the saturation and pressure plumes on a different set of permeabilities. We select the best fit in terms of RMSE and R^2-score over an ensemble of cases. We were able to achieve good accuracy (R^2 in the order of 88% for testing and over 95% for training) as dhown in figure 6. We show how performance degrades as we remove training samples and how we can alleviate some of these issues using regularization.

5.1. Experiments

We have run extensive experiments on various hyperparameters of the model. We use Tesla P100 GPU for most of our hyper-parameter search. Every run takes about 10 minutes and we are able to perform 400 using the credits allocated for the class. We monitor both the root mean square error (RMSE) and the R^2-score of calculated pressure and saturation at randomly selected points outside of the training space. We also add binarized saturation fields and complemented this loss with a segmentation loss in order to capture the non linear saturation front.

$$R^2 = 1 - \frac{\sum_{i=1}^{N} |\mathbf{y}^i - \hat{\mathbf{y}}^i|_2^2}{\sum_{i=1}^{N} |\mathbf{y}^i - \bar{\mathbf{y}}|_2^2} \tag{4}$$

Where $\mathbf{y}^i$ is the true pressure or saturation coming from ECLIPSE simulation, $\hat{\mathbf{y}}^i$ is the output from our CNN model and $\bar{\mathbf{y}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{y}^i$. A R^2-score close to 1 corresponds to a good surrogate model. We also monitor the RMSE for training and test error convergence.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \|\mathbf{y}^i - \hat{\mathbf{y}}^i\|_2^2} \tag{5}$$

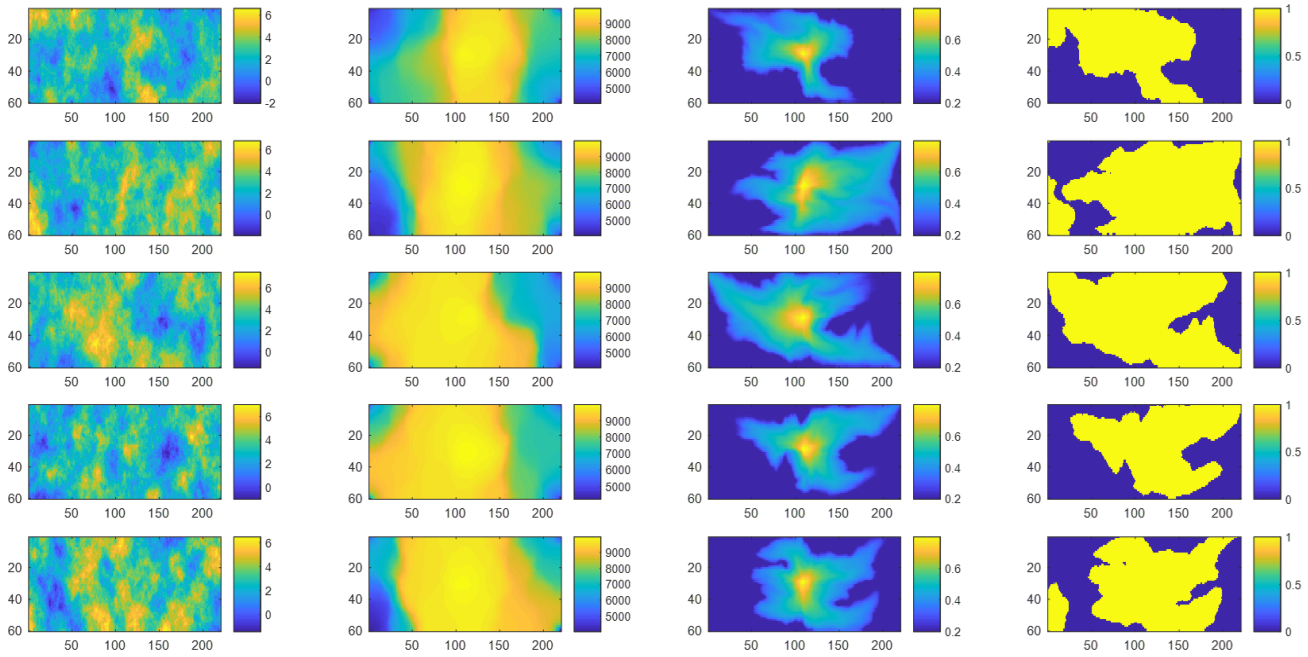

Figure 2. Five examples of input and outputs, Each column indicates log permeability (input), pressure, saturation, and binary variable ξ. For outputs, the time is fixed to t = 200 days.

As recommended in [10], we add a binary cross entropy loss in order to improve the training performance and introduce more physics to the model. It is defined as:

$$BCE = \frac{1}{NHW} \sum_{i=1}^{N} \sum_{j=1}^{HW} \|\xi(\hat{S}_{wij}) - \xi(S_{wij})\| \quad (6)$$

Where H and W are the height and width dimension of the filed (60 and 220 in our case), ξ is a binary transformation that is equal to 1 if the saturation exceeds conate water saturation and 0 otherwise. This additionnal loss has proven to improve training performance compared to our milestone case. We scale the loss with a factor λ and test different values in our hyper-parameter analysis. The final loss is:

$$LMSE = RMSE + \lambda BCE \quad (7)$$

Illustrations of the quantities of interest is represented in Figure 2. The quantities $\mathbf{y}$ represent pressure and saturation depicted in the two middle columns of Figure 2 while $\xi(S)$ is represented on the right.

The model features numerous hyper-parameters and we were able to isolate the ones that led to best results in terms of RMSE, BCE and R^2 score. We limit the training time to 200 epochs in order to be able to run more cases. We also use batch sizes of 512, 256 and 128 (with better overall results with 256). The number of layers per block is fixed as it is not a straightforward parameter to tune (change of dimension to take into account). The dimensions of the latent space and the time steps at which the training data was collected are fixed $(50, 100, 200, 500, 1000, 2000)$ as shown in Figure 8. We perform a random search (rather than grid search) on the following:

- Number of training examples : [200, 300, 400, 500, 600, 700, 800]
- dropout rate : [0, .1, .2, .3, .4, .5]
- learning rate : [1e-4, 5e-4, 0.001, 0.005, 0.01]
- weight decay : [1e-4, 5e-4, 1e-3]
- λ scaling : [0.005, 0.01 , 0.05, 0.1]

We present the results of the experiments in the form of box plots (Figures 3, 4, 5). This representation helps see the overall distribution of scores with respect to the hyperparameters and allows identifying trends that are statistically more sound than if we used single cases. On each box, the central mark indicates the median, and the bottom and top edges of the box indicate the 25th and 75th percentiles, respectively. The whiskers extend to the most extreme data points not considered outliers, and the outliers are plotted individually using the '+' symbol. Figure 3 shows the relative distributions of R^2-scores on a test set versus the number of epochs and training samples. These are the two most important hyperparameters according to our sensitivity analysis. They affect R^2-score both in average value and

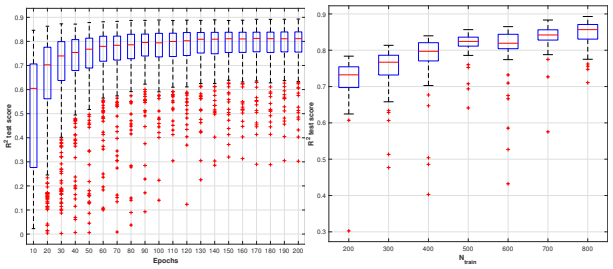

Figure 3. Box plot of R^2-scores vs number of epochs (left) and number of training sample (right) for all the experiments run.

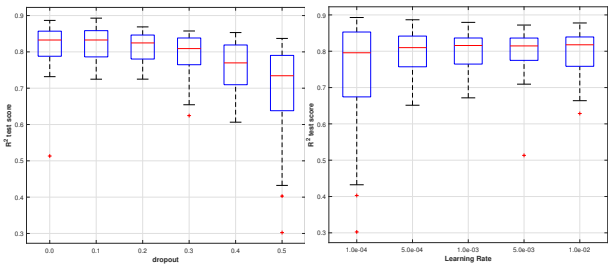

Figure 4. Box plot of R^2-scores vs dropout rate (left) and learning rate (right) for all the experiments run.

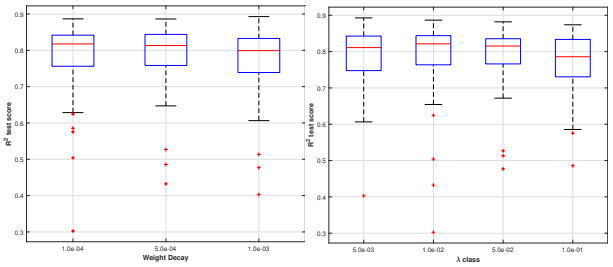

Figure 5. Box plot of R^2-scores vs weight decay (left) and contribution of the BCE loss (right) for all the experiments run.

variance. Figure 4 shows that a larger dropout does not improve the generalization error in this case and even leads to a degradation for values above 0.3. Surprisingly, the learning rate does not seem to have a large effect on the final outcome. Very small learning rate (10^{-4}) increases the spread and leads to worse answers overall. Figure 5 shows that a weight decay of 5×10^{-4} leads to the best overall results while a 0.05 weight on the BCE loss leads to slightly better results. In the next section, we look at specific results and have a more qualitative assessment of the "goodness of fit" for all these cases.

5.2. Results

Although we undestand what hyperparamters generally lead to better performances in prediction, we focus on a couple of representative cases in order to understand the quality of the output generated by our model. The best scenario in terms of testing error is obtained for the following hyperparameters:

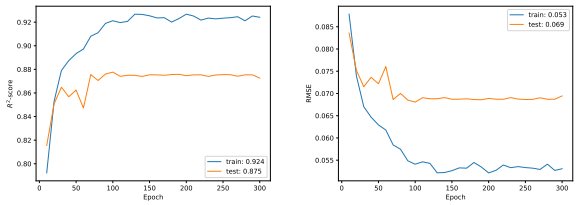

Figure 6. Evolution of R^2-score (left) and RMSE vs training epochs. Curves corresponding to training (blue) and testing (orange) are overlayed.

- Number of training examples : 800
- dropout rate : 0.2
- learning rate : 0.001
- weight decay : 5e-4
- λ scaling : 0.01

The results for R^2 and RMSE are represented in Figure 6. We see a fairly large gap between the training and testing error. More importantly, testing error seems to stagnate early on in the training process while training error continues to decrease. This indicates that we are overfitting the data in spite of a dropout rate of 0.2. A qualitative assessment of the distributions generated is presented in Figure 7 and 8.

We are able to produce surrogates of high quality using this approach. The maps shown in Figures 7 and 8 are almost identical and the relative errors we get remain in the single digit percent. Such surrogates are not common in the industry and this is a notable achievement. The downside is that we still need a large number of training sample in order to reach a good accuracy. We are interested in quantifying the degradation in quality as we remove some training samples. Figure 9 shows the The results for R^2 and RMSE for both training and test set as we remove samples. It shows a relatively smooth degradation of R^2-score as we reduce the training set size from 800 to 200. While the training error seems insensitive to the number of training sample, the test R^2-score decreases -first slowly, it remains above 85% for 500 training samples (which represents a 50-50 split in this case) but then accelerates to reach 79% for 200 training samples. In order to understand what these numbers mean qualitatively, we look at the maps produced. Getting a good surrogate for saturation is the most valuable and challenging outcome as the solution is typically quite nonlinear and can vary a lot from one permeability field to another. Moreover, a good understanding of flow path means that we are able to predict with very high accuracy the fluids fractions at the well level. We show the results obtained for 200 training samples in Figure 10.

We can identify the model with the smallest difference between training and testing scores. The best scenario in

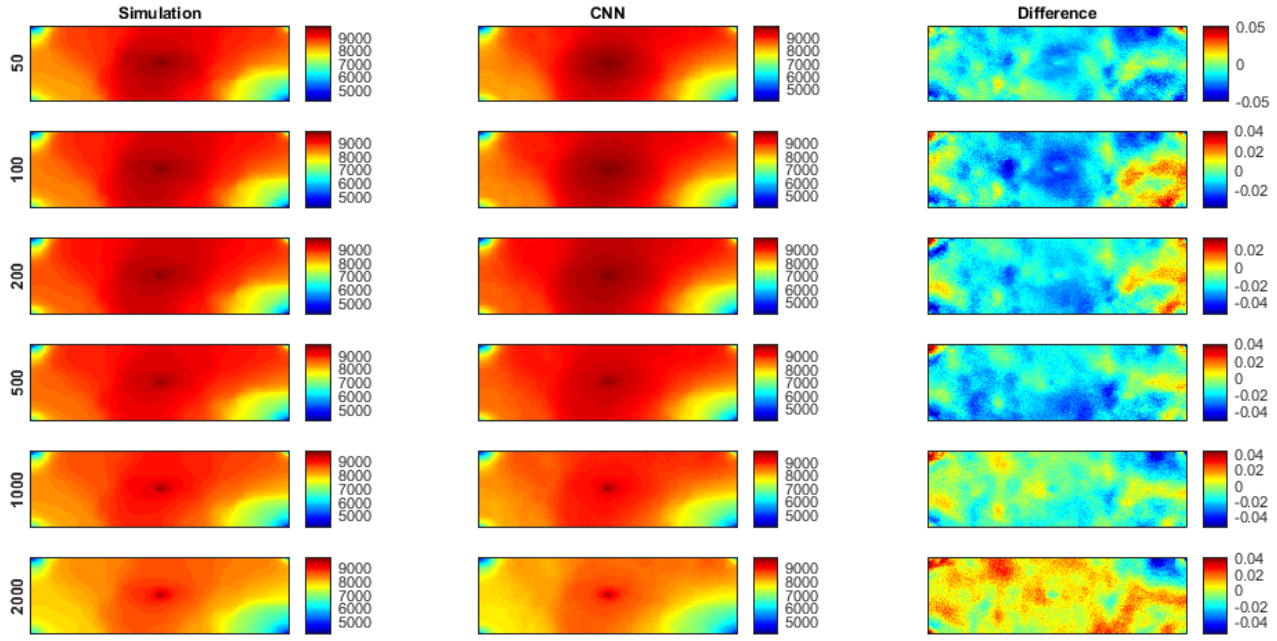

Figure 7. Pressure distribution map at different time steps for a test set. The left-most column is the pressure computed using a commercial simulator and the middle column is the result computed by our CNN. The right-most column represent the difference between the two. This model was trained using 800 samples

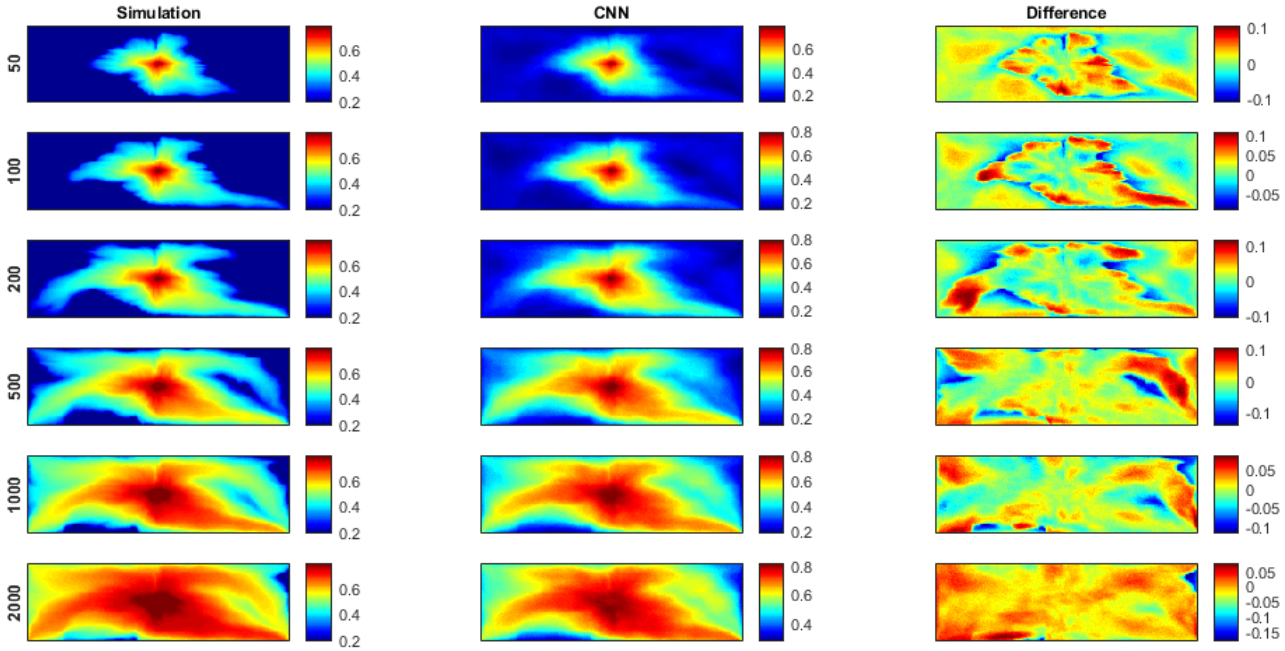

Figure 8. Saturation distribution map at different time steps for a test set. The left-most column is the saturation computed using a commercial simulator and the middle column is the result computed by our CNN. The right-most column represent the difference between the two. This model was trained using 800 samples

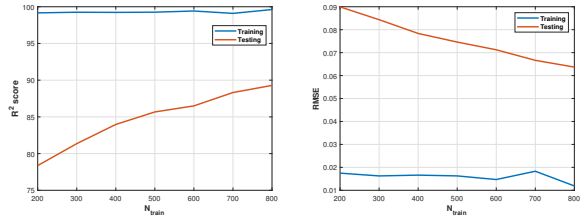

Figure 9. Evolution of R^2-score (left) and RMSE vs number of training samples. Curves corresponding to training (blue) and testing (orange) are overlayed.

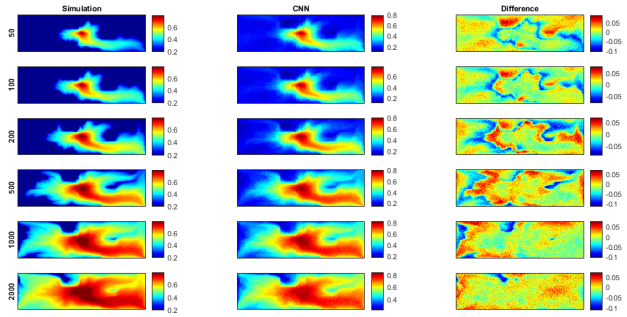

Figure 10. Saturation distribution map at different time steps for a test set. The left-most column is the saturation computed using a commercial simulator and the middle column is the result computed by our CNN. The right-most column represent the difference between the two. This model was trained using 200 samples

terms of difference betwwen training and testing error was obtained for the following hyper-parameters:

- Number of training examples : 800
- dropout rate : 0.4
- learning rate : 0.01
- weight decay : 0.001
- λ scaling : 0.1

The training curves for this case is represented in Figure 11

We see on the distributions displayed in Figure 12 the evolution of the saturation with training epochs. We see that the resolution and fidelity of these maps increases rapidly over the first 50 epochs and then stagnates as indicated in the evolution of the R^2-score plotted in Figure 6.

5.3. Discussion

Even though the RMSE and R^2-score could be further improved and the gaps between training and testing error remain quite significant, the qualitative assessment of our results shows good predictability in terms of mapping of path of preferential flow, flow barriers and breakthrough to

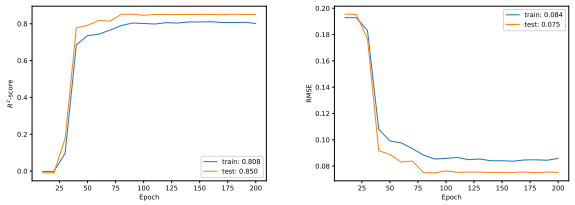

Figure 11. Evolution of R^2-score (left) and RMSE (right) vs training epochs for the case with the lowest difference between training and testing error. Curves corresponding to training (blue) and testing (orange) are overlayed.

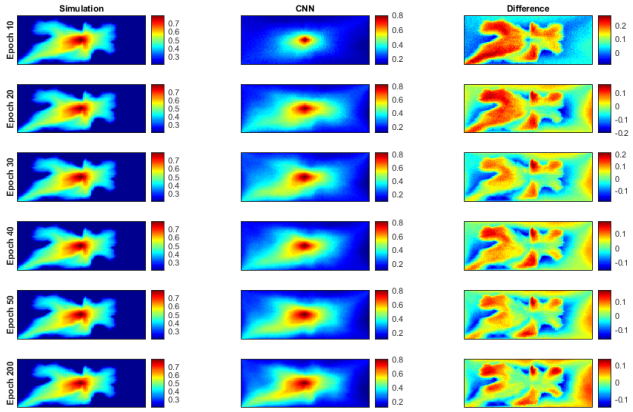

Figure 12. Saturation distribution map at 500 days for different epochs of training. The left-most column is the saturation computed using a commercial simulator and the middle column is the result computed by our CNN. The right-most column represent the difference between the two. This model was trained using 800 samples

producing wells. Such a model would be valuable to compute water breakthrough at wells. Fractional flow is one of the most non-linear -and difficult to simulate- quantity at the wells level. The exercise we overtake is to map directly the distributions of pressures and saturation in the reservoir. Most existing studies focus on the matching of saturations at given points or the production of fluids at well locations. Not only is the mapping quite accurate but most of the features are conserved for low numbers of training samples and an optimal performance is achieved with a small number of epochs (typically less than 100). As we keep training, the performance does not improve much. This indicates that training time can be cut in half without sacrificing too much in terms of performance. Figure 13 shows the compared productions of water at the four corner wells for one of the test cases. The match is very close with a RMSE error of 10^{-2}.

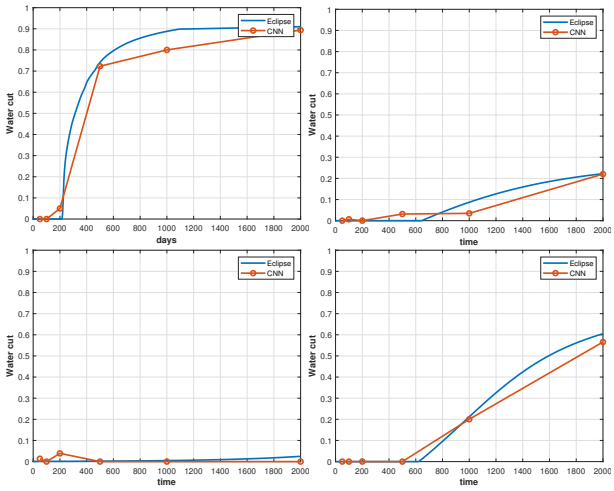

Figure 13. Evolution of watercut vs time at the 4 producer wells in one realization of the SPE10. The Eclipse simulation results are in blue while the CNN computed results are in orange circles.

6. Conclusion/Future Work

This work demonstrates the capability of CNN encoder-decoder architecture coupled with a physics based loss function to match simulation results with a high degree of accuracy (low RMSE, high R^2). The network is able to solve for the spatial and time evolution of the saturation and pressure. From Section 5, we are able to conclude that a large data set with some regularization was able to perform well for this problem. Dropout has a regularizing effect on the model up to a certain point and leads to the discrepancy between the training error and test error to decrease. The learning rate/weight decay on the other hand do not seem to have an effect on the model. This is certainly related with the fact that most of the learning occurs during the first few epochs. It could also be explained by the reduced size of our dataset with a few hundred "images" used to calibrate the regression model. This is made possible by our engineered loss function that instills more physics to the model parameterization. The effect of λ scaling demonstrates this. A starter code package has been uploaded to: `https://drive.google.com/open?id=14TZYsRzcLx6f99IGbho_3puE9NUYiulv`

In the future we would like to investigate the following:

- Further study the λ scaling and try new features (derivative based)
- Learn more time steps/predict more time steps.
- Learn a 3-D system (3-D convolutions will be needed)
- Learn scenarios with varying well placement
- Expand the loss function to include other physics (gravity, capilarity)
- Learn using partial/missing data or introduce noise to the data
- Deduce oil production from images/use oil production as a feature

7. Contributions and Acknowledgements

Every team member contributed to nearly every task of the project. Each team member was extensively involved in the 3 most laborious tasks of this project:

- QC the hdf5 data manipulation
- Debugging the network code
- Changing the network for the purposes of this study

This section lists the most distinguishing contributions by each member.

7.1. Cedric Fraces Gasmi

- Training runs
- Conducted the most through literature review and found the most promising network architecture
- Designing and running the hyper-parameter search experiments

7.2. Jihoon Park

- Generated training data by completing thousands of SPE10 simulation runs
- Organized the data into a usable hdf5 files
- Modified the neural network architecture for the project

7.3. Dong Hee Song

- Modified training, test, and output data sets
- Training runs on google cloud compute
- Analyzing the output files of the hyper-parameter search experiments

7.4. Github Repository and Libraries

The starting point of this project was the work done by Zabaras *et al.* [10]. The original Github repository which was the starting point of our network archeticture can be found at: `https://github.com/cics-nd/dcedn-gcs`.

This project incorporated the following libraries: PyTorch [11], Matplotlib/Seaborn [6], NumPy [3], and scikit-learn [12].

8. References/Bibliography

References

[1] J. J. Arps et al. Analysis of decline curves. *Transactions of the AIME*, 160(01):228–247, 1945.

[2] F. Brons. On the use and misuse of production decline curves. *Producers Monthly*, 27(9):76–70, 1963.

[3] N. Developers. Numpy. *NumPy Numpy. Scipy Developers*, 2013.

[4] M. Fetkovich et al. A simplified approach to water influx calculations-finite aquifer systems. *Journal of Petroleum Technology*, 23(07):814–828, 1971.

[5] M. J. Fetkovich et al. Decline curve analysis using type curves. *Journal of Petroleum Technology*, 32(06):1–065, 1980.

[6] J. Hunter, D. Dale, and M. Droettboom. Matplotlib. *The Architecture of Open Source Applications*, 2:165–178, 2011.

[7] A. Karpatne, W. Watkins, J. Read, and V. Kumar. Physics-guided neural networks (pgnn): An application in lake temperature modeling. *arXiv preprint arXiv:1710.11431*, 2017.

[8] L. W. Lake. *A generalized approach to primary hydrocarbon recovery*, volume 4. Elsevier Science Limited, 2003.

[9] C. C. Mattax, R. L. Dalton, et al. Reservoir simulation (includes associated papers 21606 and 21620). *Journal of Petroleum Technology*, 42(06):692–695, 1990.

[10] S. Mo, Y. Zhu, N. Zabaras, X. Shi, and J. Wu. Deep convolutional encoder-decoder networks for uncertainty quantification of dynamic multiphase flow in heterogeneous media. *arXiv preprint arXiv:1807.00882*, 2018.

[11] A. Paszke, S. Gross, S. Chintala, and G. Chanan. Pytorch. *Computer software. Vers. 0.3*, 1, 2017.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[13] M. Raissi. Deep hidden physics models: Deep learning of nonlinear partial differential equations. *The Journal of Machine Learning Research*, 19(1):932–955, 2018.

[14] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.

[15] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10566*, 2017.

[16] Schlumberger. Eclipse industry-reference reservoir simulator.

[17] Society of Petroleum Engineers. SPE Comparative Solution Project.

[18] Y. Zhu, N. Zabaras, P.-S. Koutsourelakis, and P. Perdikaris. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *arXiv preprint arXiv:1901.06314*, 2019.