CS230

Deep Learning Model for Subsurface Flow Prediction with Multifidelity Data

Yusuf Nasir
Department of Energy Resources Engineering
Stanford University
nyusuf@stanford.edu

Abstract

In order to maximize value from a petroleum or geothermal reservoir, the time-varying rates at which fluids are produced and injected needs to be optimized. This typically requires thousands of expensive flow simulations. In this work, we used a convolutional neural network (CNN) to condition a long short-term memory (LSTM) model in order to predict flow from the subsurface using multifidelity data. This entails generating a lot more training data at low fidelity level and correcting the error incurred by the use of low fidelity data with few high fidelity data. A speed up of about 7 times was achieved compared to use of only high fidelity data. This is a significant time saving when you consider that each field scale simulation could take hours to days.

1 Introduction

A brute force approach to solving the field development optimization problem entails discretizing the bound of possible well pressures to be imposed on each well into different regions and simulating all possible combination of these regions. The optimal decision will then be the well rates (obtained after flow simulation with the well pressures) that give the highest value of an economic metric under consideration. However, due to the significantly high number of possible well pressure combination in cases where large number of wells are considered, the brute force approach is not feasible. Traditional optimization algorithms, such as particle swarm optimization, genetic algorithm, have been considered, and they generally require many flow simulations in order to solve this complex optimization problem. Each of this field-scale simulation can take hours to days, making it infeasible or highly computational expensive to solve this problem using the actual flow simulator.

In this work, we consider the use of convolutional neural network (CNN) and long short-term memory (LSTM) to develop a deep learning model to replace the time-consuming flow simulation process. Due to the uncertainty in our knowledge of the subsurface, we considered different representations (realizations) of the the geological model. This can be mathematically represented by:

$$\hat{q}^{(i,j)} = f(m^{(i)}, x^{(j)}) \tag{1}$$

where f is the deep learning model we seek to find, that given a realization $m^{(i)}$ and a vector of well pressures $x^{(j)}$ predicts the flow rate $\hat{q}^{(i,j)}$ as a function of time.

Because it is relatively more computationally to simulate the high fidelity model (m_h), we use a much larger amount of low-fidelity data which are inexpensive to obtain from a low dimensional form

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

of the realizations (m_l) for flow prediction and few high-fidelity data to correct the error incurred due to the use of low fidelity data. We train two deep learning models: the first model uses the low fidelity data to map from the input (m_l, x_l) to the output (q_l) while the second model learns the error or mapping from q_l to q using a few high fidelity data (m_h, x_h) and an embedding extracted from the first model (ξ). This is further discussed in the methodology section.

2 Related work

Machine learning techniques such as support vector regression (SVM) Guo and Reynolds [2018] and gradient boosting (Nwachukwu et al. [2018] and Nasir et al. [2019]) have been employed to develop surrogates for the flow simulator. However, in these studies the predictor was a scalar value (the cumulative rate) while in our study we are interested in time-series prediction. For this reason, deep learning techniques have been employed for time-series prediction in a number of related studies.

More recently Tang et al. [2019] have used a combination of CNN and LSTM for flow rate prediction.In Tang et al. [2019], however, a fixed well pressure was imposed on the wells and the goal was to predict flow rate for only varying realizations of the geological model. Jin et al. [2019] have also employed deep learning technique for flow rate prediction for varying well pressures but fixed realization. In all the aforementioned work, the training data was generated using the high fidelity geological models.

Meng and Karniadakis [2019] proposed the use of a composite neural network that learns from multifidelity data. They used three neural networks (NNs), with the first NN trained using low fidelity data and coupled to two high fidelity NNs, one with activation functions and another one without, in order to discover and exploit nonlinear and linear correlations, respectively, between the low fidelity and the high fidelity data. In contrast to the approach in Meng and Karniadakis [2019] which uses only fully connected layers with scalar inputs, a composite CNN and LSTM is used in our study to predict time-series data, which is the evolution of flow rates as a function of time.

3 Dataset and Features

The required dataset for training, evaluation and testing was generated using Stanford's automatic differentiation general purpose research simulator (AD-GPRS). The high fidelity realizations (m_h) of the geological models of grid size 120 x 120 are coarse-grained using an upscaling process to generate the low fidelity realizations (m_l) of grid size 20 x 20 as shown in figure 1.The steps taken to generate the data are as follows:

Step 1: Sampled 400 random well pressures (x_l) from the allowed well pressure bound. $x_l \in \mathbb{R}^{36X400}$, where 36 represent the well pressures to be imposed on 6 wells for six different time intervals.

Step 2: Simulate all 400 pressure samples using the reservoir simulator with 50 low-fidelity realizations (m_l) (20,000 flow simulations). This took about 7 hours. After flow simulation, $q_l)$ is extracted which has five outputs in each time interval.

Step 3: Used k-means clustering, with dissimilarity (Euclidean distance) calculated using q_l, to select 50 representative well control pressures ($x_h \in \mathbb{R}^{36X50}$) from the 400 generated in step 1, and ran the flow simulation (2500 simulations) using the 50 high-fidelity realizations (m_h). This took about 41.5 hrs. After flow simulation, q_h is extracted, which is the quantity we are interested in predicting.

The permeability field depicted in figure 1 defines the conductivity of fluid in different parts of the reservoir. Min-max scaling was used to normalize the permeability and well pressure values before training. The 20,000 low-fidelity data was divided into 18,000, 1000 and 1000, while the high fidelity data was divided into 2000, 250, and 250 for training, evaluation and testing, respectively.

4 Methods

Figure 2 shows the architecture of the first deep learning model used to predict q_l. Because of the spatial and high dimensional nature of the geological models m_l, CNN was used to find a lower dimensional representation ξ_l for each of the 50 upscaled models m_l. ξ_l serves as input to two fully

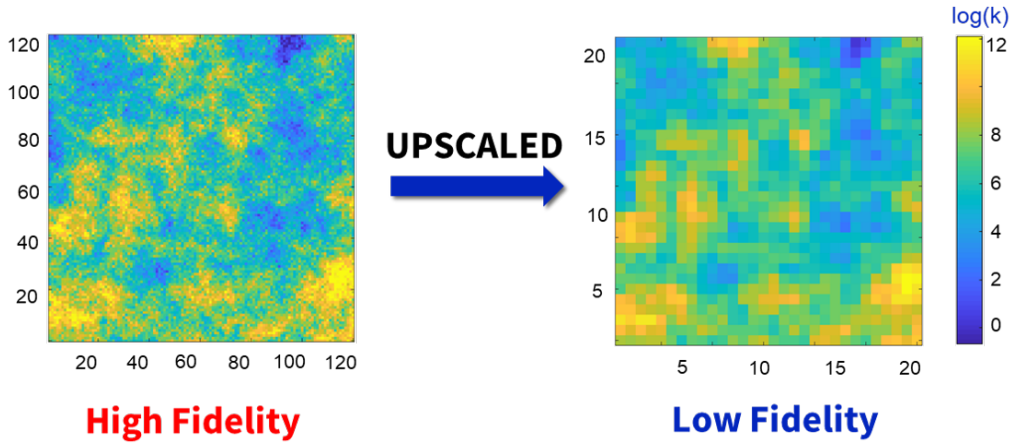

Figure 1: Log permeability field for a high-fidelity realization of size 120 x 120 and an upscaled low-fideity realization of size 20 x 20

connected layers (FC_1 and FC_2) that are used to initialize the cell and hidden state of the LSTM. The LSTM also take the well pressures x_l as an input with six time intervals. The output of the LSTM is the predicted flow rate q_l. Remember that we are interested in predicting the flow rate of the high-fidelity realizations q_h, hence we used a second deep learning network to model the error $(q_h - q_l)$.

In the second deep learning model (figure 3), a CNN model is used to find a low dimensional representation ξ_h of the high-fidelity realizations m_h. ξ_h and ξ_l (extracted using m_l and x_h as inputs in the first model) are then concatenated and used as inputs into two fully connected layers (FC_1 and FC_2) that are used to initialize the cell and hidden state of the LSTM in the second model. The LSTM also takes as input the predicted q_l (extracted using m_l and x_h as inputs in the first model) and x_h. The output of the second model is q_h. All inputs to the second model are normalized using min-max scaling.

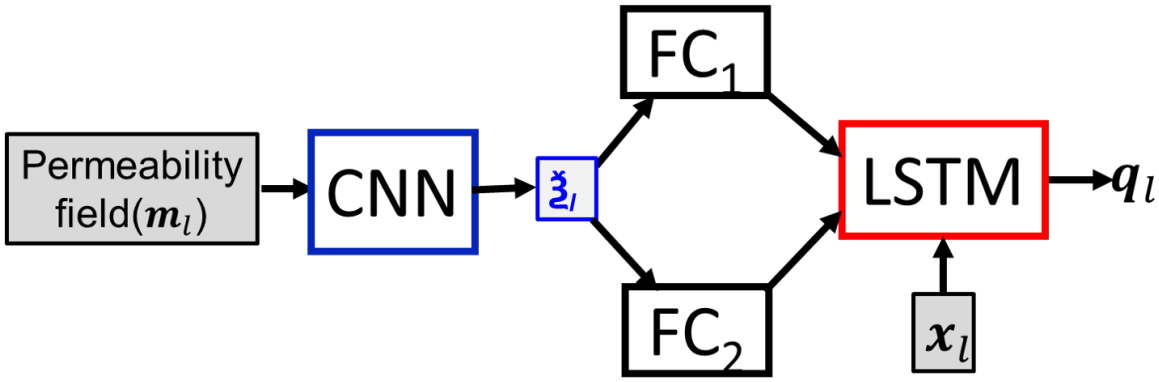

Figure 2: Deep learning architecture that utilizes low-fidelity data

The composition of the CNN used in this work is same as that in the encoding model of Tang et al. [2019]. It has 4 2D convolutional blocks and 3 resnet blocks. The LSTM comprises of six time intervals with 200 neurons in both deep learning models. The mean absolute error (equation 2) was used as the loss function because it provided lower test error compared to the mean squared error.

$$L_l = \frac{1}{N_t}\sum_{i=1}^{N_t}\left|\hat{q}_l^{(i)} - q_l^{(i)}\right|, L_h = \frac{1}{N_t}\sum_{i=1}^{N_t}\left|\hat{q}_h^{(i)} - q_h^{(i)}\right| \tag{2}$$

where N_t is the length of q_l and L_l and L_h are the loss functions for the first and second deep learning models, respectively.

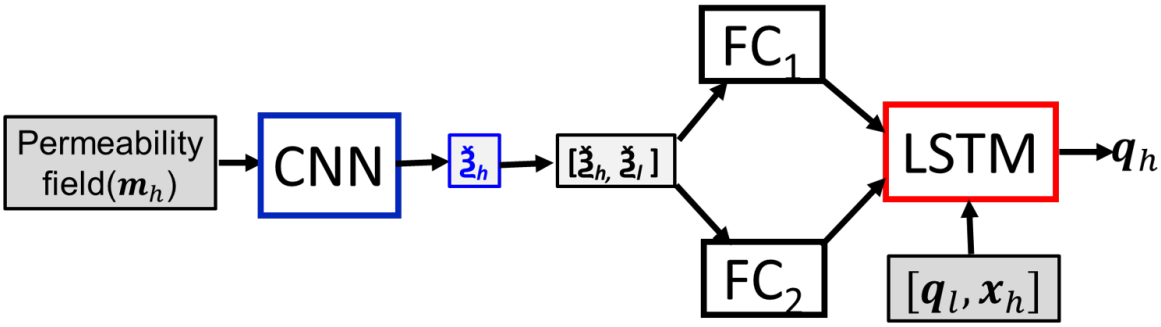

Figure 3: Deep learning architecture that utilizes high-fidelity data

5 Results and Discussion

Table 1 shows the optimal hyperparameters obtained using a grid search process.

Table 1: Optimum hyperparameters

Hyperparameters	Values
Learning rate	0.03
Number of epochs	250
Batch size	16
Dropout rate	0.2

Adam optimizer was used with the $beta$ parameters set at their default values and the optimal learning rate as shown in Table 1 is 0.03 with a decay rate of 0.02. It was noticed that the lesser the batch size the better the trained model was able to generalize, however, to reduce the training time, a batch size of 16 provided a good balance between training cost and generalization.

Table 2 shows the mae during training,evaluation and testing. Using low batch size and monitoring the validation loss ensured the models did not over fit.

Table 2: Mean absolute error (MAE) for the different model

Model	Training MAE	Dev MAE	Test MAE
Low fidelity model	77	82	74
High fidelity model	65	72	68
Third model(only high-fidelity data)	107	111	115

Figure 4 shows the results for a single well pressure and realization predicted by both deep learning models and a third model trained on only high-fidelity data.Figure **??** shows the actual and predicted q_l by the first deep learning model. There is a very close agreement between the two curves. This is supported by the low mean absolute prediction error. In figure **??**, the predicted q_l and q_h values are compared with the actual q_h curves and it is evident that a significant amount of error (difference between red curve and black dots) is incurred due to the use of the low-fidelity. However, the second deep learning model was able to correct this error using few high fidelity training data. In figure **??**, the green curve represent a third deep learning model that is trained using only the few high-fidelity

data (m_h and x_h) to predict q_h using the setup in the first deep learning model.Due to only few training data for varying realization and well pressures, it was not as accurate as the case where both low and high-fidelity data were utilized.

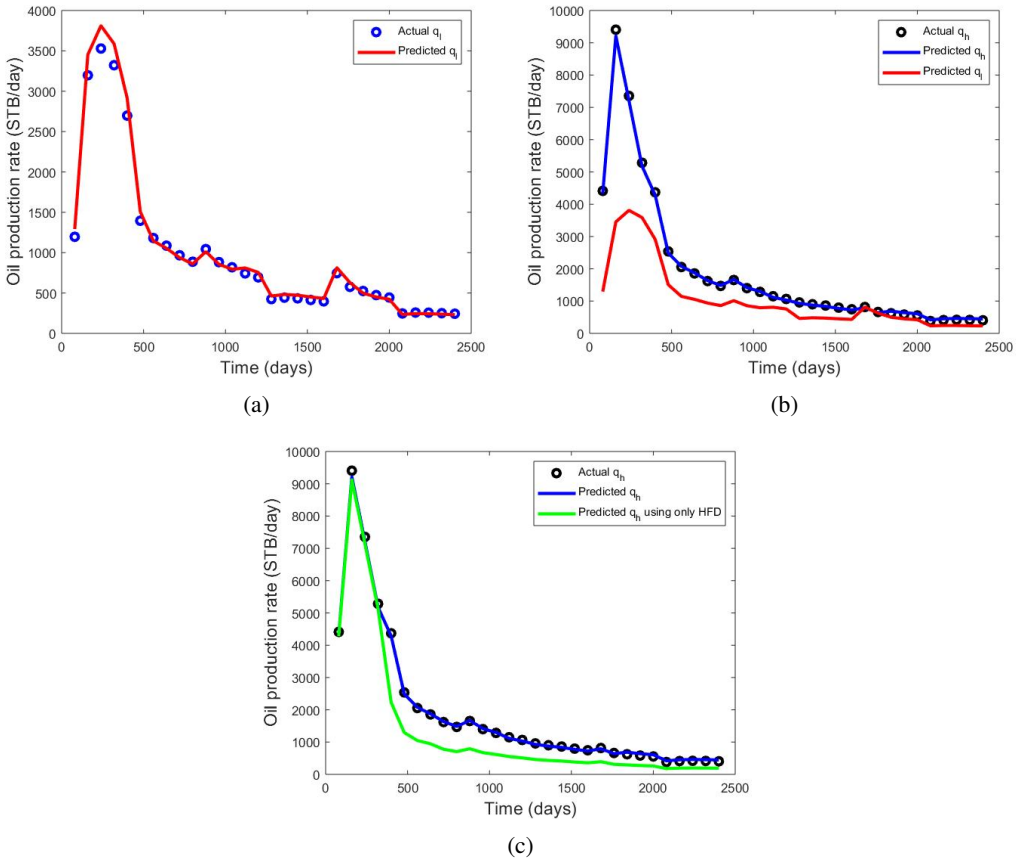

Figure 4: Results for a single well pressure setting and realization predicted by both deep learning models and a third model trained on only high-fidelity data

6 Conclusion and Future Work

In this work, we have demonstrated deep learning models can be used for the highly nonlinear subsurface flow prediction with a combination of low and high-fidelity data. This allows us to generate a large amount of low-fidelity data at a relatively lower computational cost and few high-fidelity data to correct the error introduced by the low-fidelity geological realizations. Assuming we are to simulate all the 50 realizations and 400 well pressure samples, that would have taken 332 hrs, however, we generated a combination of low and high-fidelity data in 48.5 hrs resulting in a speed up of about 7.

In this work only the oil rate was predicted,however, in optimization, the water production and injection rate are relevant quantities to calculate the economic value to be derived from a certain well pressure setting. Hence it will be important to extend this work to predict water production and injection rates. This model can also be deployed as a surrogate during production optimization under uncertainty.

7 Acknowledgement

I will like to thank Yimin Liu, Meng Tang and Yong Do Kim for the useful discussion on deep learning architectures and training for problems within the domain of subsurface flow prediction. I

will like to acknowledge Stanford CEES for the GPU resources provided to train the deep learning models.

8 Code

The code for this work can be found at https://github.com/yus-nas/Composite-DNN-for-Well-Production-Optimization

References

References

Z. Guo and A. C Reynolds. Robust life-cycle production optimization with a support-vector-regression proxy. *SPE Journal*, 2018.

Z. L. Jin, Y. Liu, and L. J Durlofsky. Deep-learning-based reduced-order modeling for subsurface flow simulation. Technical report, 2019.

X. Meng and G. E. Karniadakis. A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse pde problems. *arXiv preprint arXiv:1903.00104*, 2019.

Y. Nasir, W. Yu, and K. Sepehrnoori. Hybrid derivative-free technique and effective proxy treatment for constrained well placement and production optimization. *Journal of Petroleum Science and Engineering*, page 106726, 2019.

A. Nwachukwu, H. Jeong, A. Sun, M. Pyrcz, and L. W. Lake. Machine learning-based optimization of well locations and wag parameters under geologic uncertainty. In *SPE Improved Oil Recovery Conference*. Society of Petroleum Engineers, 2018.

M. Tang, Y. Liu, and L. J. Durlofsky. A deep-learning-based surrogate model for data assimilation in dynamic subsurface flow problems, 2019.