Imperial College London

Department of Earth Science and Engineering

MSc in Applied Computational Science and Engineering

Independent Research Project
Project Plan

Deep learning proxy of a reservoir simulation

by

Raha Moosavi

seyedeh.moosavi19@imperial.ac.uk
GitHub login: acse-srm3018

Supervisors:

Dr. Daniel Busby
Prof. Yves M. Leroy

August 2021

GitHub repository: https://github.com/acse-2020/acse-srm3018

Abstract

Field development and reservoir management require a large number of multi-phase flow simulations. Artificial neural networks (ANNs) offer a hopefully faster approach where the data is used for training. In this project, our objective was to use a combination of a convolutional neural network (CNN) model (residual U-Net) with a long short-term memory (LSTM) to approximate the 2-D reservoir properties (saturation and pressure). For this purpose, we used a synthetic dataset (3000 data results from simulation) generated using the industrial software Intersect. Two deep learning-based models with various architectures were trained for pressure and saturation, each model including different blocks and layers. The first model is Recurrent Residual U-net and the second one is Attention Recurrent Residual U-net. After training, 750 cases have been used as a test dataset to assess the proxy models performance with unseen data. Although the accuracy of both models for predicting pressure and saturation in testing cases are high, the accuracy of the A RR U-net is better with 0.2% lower mean relative error. This difference is due to added attention blocks which help the network to learn from the most useful and relevant information.

1 Introduction

Forecasting subsurface flow is highly important for oil and gas reservoir management and field development optimization. A precise model is required for simulating fluid flow on a 3D grid in a reasonable time. The common numerical modelling method of the reservoir fluid flow includes the finite-difference (FD) (Ertekin et al. 2001), the finite volume (FV) (Alpak et al. 2010) and the finite element (FE) (Fung et al. 1992) methods. These methods require a discretized grid which size defines the precision of the prediction. Also, they rely on simplifying assumptions of the exact physics of the real world.

An accurate fluid flow simulation through porous media using numerical simulation methods becomes quickly time-consuming (several hours for a single run) and hundreds to thousands of runs are necessary to calibrate the prediction and its uncertainty using observation data. Therefore, a major research effort has been conducted over the last 20 years for constructing proxy models to subsurface flow simulators (Mo et al. 2019, Liu et al. 2019, Zhou 2012, Vo & Durlofsky 2015, Jin et al. 2020). Artificial Neural Network (ANN), is one of the most emerging technology recently (Karumuri et al. 2020, Jin et al. 2019, Tang et al. 2021, Padmanabha & Zabaras 2021). it has entered into the field of scientific computing and is set to become the new paradigm in modelling and computation. The aim of using ANNs is to allow the system to learn and to train from the data rather than to create and solve a system of equations. The deep learning (DL) based models, which are the ANNs with multi-layers between inputs and outputs, have been also applied to model fluid flow (Zhao et al. 2018, Wang & Lin 2020, Cheung et al. 2020).

Recent successful application of DL for processing images using high-dimensional data (Yoo et al. 2017) and natural language processing have led to conducting more research on creating a DL proxy model for a nonlinear system of high dimensional. The high dimensional system considers as the system with higher features than the number of observations. A fully Convolution Neural Network (CNN) has been applied by Zhu and Zabaras (Zhu & Zabaras 2018) for predicting single-phase steady-state fluid flow firstly (Goldberg 2016). Jin et al. (Jin et al. 2019) created a DL proxy model to prove that it can recognize the high dimensional underground system. Tang et al., (Tang et al. 2020) developed a DL proxy model to predict dynamic flow in the channelized geological system. The proxy model is a combination of CNN which has several convolution layers (filters) to capture various features. long short-term memory (LSTM) network (Sainath et al. 2015) employs to capture temporal dynamics in the system (Graves et al. 2013). Residual U-Net is a deep conventional model

(Tang et al. 2020) which has been applied for biomedical image segmentation before (Alom et al. 2019).

In this study, fully CNNs (Jin et al. 2017) is proposed for scaling the proposed model to a total oilfield. Therefore, we proposed to employ DL based model to create an efficient proxy of the 3D fluid flow simulator. The objective is to train a DL model that will take 2D cells of permeability and porosity values as an input and then produce 2D images of saturation and pressure as a function of time with given discretization.

Since deep CNNs have some properties including effectiveness for high dimensional problems, they can potentially be used as a proxy model to capture the spatial and temporal features of the reservoir and improve the computational cost. Therefore, investigating the reliability of DL models to use as proxy models for reservoir simulation is the main objective of this project. We trained and evaluated two model architectures including recurrent residual U-net (RR U-Net) and Attention recurrent residual U-net (A RR U-Net). Both architectures use a U-net autoencoder as their main structure of model. U-net architecture is a type of u-shaped CNN that started wit contraction (encoded) followed by expansion (decoded) layers to extract the feature of geological images (permeability) and state variable map (saturation and pressure map). ConvLSTM (Swapna et al. 2018) is used in combination with residual recurrent blocks and attention blocks to extract the system progression with time.

Additionally, we intended to relate the performance of our proxy model to network structures that have been used in this study. Summary of the research goals was:

- Applying deep neural networks to simulating reservoir pressure and saturation by feeding geological parameter map (permeability) to network.
- Assessing the surrogate model performance in simulating reservoir properties (pressure and saturation).
- Investigate the proposed method (A RR U-Net) performance in comparison to the the surrogate models used in previous studies (RR U-Net).

This report structure summarised as follows. The software description section provides the methodology and the neural network model layers and architectures. The GitHub repository and its requirements are first presented before defining the training process including the dataset, its preprocess and the loss function. The result section sees the applications of the two models to datasets including the two-phase flow through a channelized system. The predicted pressure and saturation maps obtained with the neural networks are compared with the data and their differences and the relative error are presented. A summary of this work and the suggestion for future work are found in the final conclusion.

2 Software description

In this section, the models' architecture, the type of blocks and layers, sequence of layers in blocks and sequences of blocks in the network have been described. The code metadata including the frameworks, libraries, system, etc. has also been presented.

2.1 Deep learning model architecture

U-net, an encoder-decoder architecture, introduced by Olaf Ronneberger et al. (Ronneberger et al. 2015), has been applied successfully to many problems. Firstly, It was applied to the remote sensing image analysis field for road extraction. After that, it was adopted by researchers for other applications such as medical image segmentation. In this study, we used two models based on U-Net architecture, recurrent residual U-net (RR U-net) and attention recurrent residual U-net (ARR U-net). These are DL based models benefit from the advantages of attention, residual and RNN blocks together in their U-net network. Residual block helps the training process by preventing vanishing gradient problems, easier training process by better providing backpropagation. Attention block contributes in concentrating on the most similar input sequence for each output. This has been done by creating a path to the inputs directly. Another advantage of the attention block like the residual block is reducing the vanishing gradient problem.

Convolutional LSTM uses a recurrent block to capture the sequence data features (temporal features). It is a type of LSTM (Long Short-Term Memory) where we add a convolution block inside the LSTM cell.

2.1.1 Encoder-decoder architecture

The proposed architectures have encoding, decoding and a bridge part that connects these two parts. U-Net is used to capture spatial features and extract the complicated input and output maps' relationships. The U-Net uses convolution together with the batch normalization and a ReLU activation function. In the residual block, these layers are replaced by a pre-activated residual block. Also, attention blocks were just added before concatenating encoder and decoder tensors in the decoder.

Fig.1 shows the models architecture, where the encoder takes permeability map as input and passes it through different encoder blocks. This contributes in learn local features by the network. The encoder consists of four encoder blocks, which are built using the pre-activated residual block. The output of each encoder block is then copied and concatenated for the corresponding decoder part. In the case of A RR U-Net model, attention blocks were added to the decoder part before concatenation with corresponding encoder part.

The first and third convolution layers in the encoder uses a stride of 2 in the second and the third encoder block. This helps spatial dimensions (height and width) of the feature maps to reduced. A stride value of 2 reduces the spatial dimensions by half.

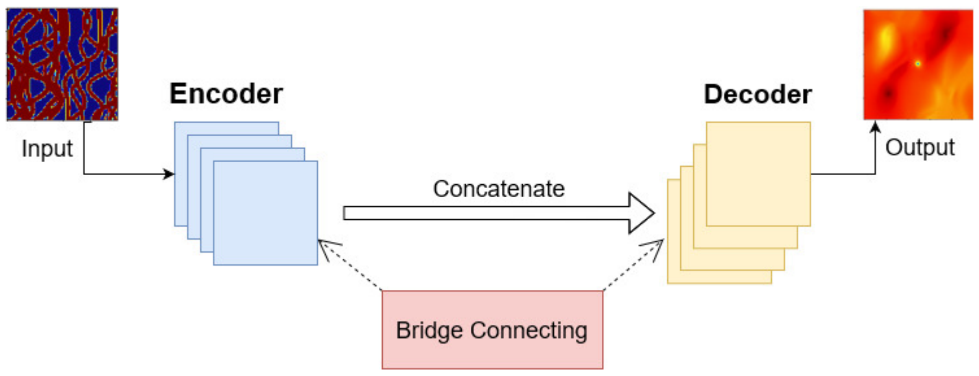


Figure 1: Recurrent Residual U-net including encoding and decoding.

On the other hand, the decoder part takes the feature maps from the bridge and the encoder blocks. It extracts local features which are then used to generate the final output. The decoder consists of four decoder blocks, and after each block, the spatial dimensions of the feature maps are doubled. As a consequence the number of feature channels are reduced. Each decoder block begins with upsampling, where we use the transposed convolution block to increase the spatial dimensions. The upsampling layer is utilized to make the spatial dimensions of the feature maps doubles. These feature maps are then concatenated with the appropriate connection from the encoder block. These connections help the decoder blocks to get the local feature learned by the encoder network. After this, the feature maps from the concatenation operations are passed through a pre-activated residual blocks.

2.1.2 Neural network layers

As it is mentioned in the previous section, we used a different block in U-net architecture to create proxy models for accurate reservoir simulation. These blocks are listed below:

1. Convolution blocks
2. Residual blocks
3. Recurrent blocks
4. De-convolution blocks
5. Attention blocks

Two main models in this study are:

1. RR U-net: the base of this model is U-net and recurrent convolution block and residual block used in this model. the sequence of this layer in this model has been shown in fig2.
2. Attention RR U-net: this model is the similar to RR U-NET model, the only difference is that the attention blocks have been added to decoder blocks to better spatial feature reconstruction.

Each block consists of a sequence of layers. Different blocks, their layers, the sequence of layers in each block and the sequence of blocks in models have been shown in figure 2. Following the reason behind each block and its layers are discussed in more details.

- **Convolution blocks**

  Convolution, batch normalization and ReLU activation function are the three layers used in this block sequentially. The convolution layer is the main layer of the convolution block, which enables networks to extract high-level features such as edges, lines, curves, etc from the input tensor. The convolution layer consists of the mathematical convolution operation, which takes two inputs, the input tensor and a small kernel matrix. The batch normalisation is a regularizer that is widely employed in CNN models to reduce internal covariance shift which helps in creating deeper, faster and more stable networks by normalizing the input layer. The activation layer introduces non-linearity in the neural network. ReLU (Rectified Linear Unit) is the most popularly used activation function in CNN. It is used in all the layers except the output layer. The activation function of the output layer depends upon the type of problem we are trying to solve. The sequence of CONV-BN-ReLU helps to enhance the network performance by improving generalisation.

- **Residual recurrent blocks**

  The residual block consists of series of convolution layers along with a shortcut (identity mapping) connecting both the input and the output of the block. This identity mapping helps in better flow of information, i.e., directly from one layer to another, bypassing the convolution layers. This bypassing helps in better flow of gradients during the backpropagation.

- **De-convolution block**

  In this block, we used a sequence of transposed convolution and then a convolution block (conv2D, BN, ReLU). Transposed convolution is a process that takes the transpose of the kernel matrix and reverses the convolution process. Transposed convolution result in increasing the output size while convolution blocks decrease the size of the output.

- **Convolutional LSTM 2D**

  Convolutional LSTM is a variant of LSTM which uses convolution operation in the input transformations and recurrent transformation in the LSTM block. The Convolutional LSTM

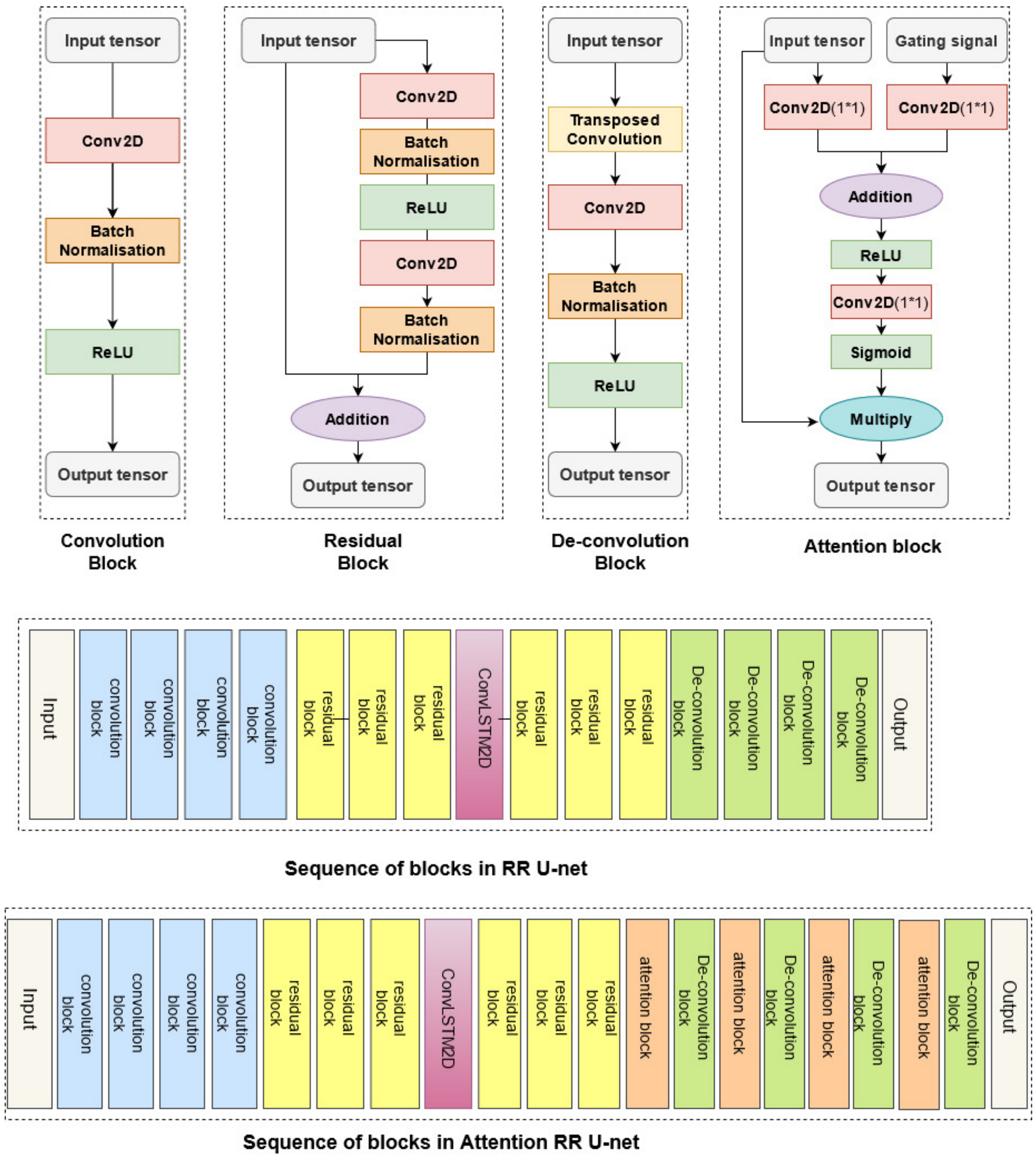


Figure 2: blocks and their layers and sequences of layers in blocks

usually applies to time series of images. For example, for weather forecasting by taking the time series of sensor values of 2D grids as input.

- **Attention blocks**

  As mentioned before, the encoder process extracts features and the transposed convolution block is used for upsampling to increase the size to the original one before concatenating them. It is expected that network is able to accurately reconstruct the input while the decoder process results in eliminating spatial features and information of input images like the edges, contours and position components. The attention block helps in interacting with spatial points, enhancing the key areas, and restraining irrelevant areas by paying attention to the important regions.

2.2 Code metadata

2.2.1 Platform

The language used for developing software is Python3. For the developing platform, the software is developed under Windows 10. The libraries and frameworks are also described. The platform for machine learning is Google Colab, it provides a NVIDIA-SMI 470.42.01 Driver Version: 460.32.03, CUDA Version: 11.2 with 27.3 Gigabytes RAM . The user can use one GPU up to 12 hours per session for free by a Google account.

2.2.2 Libraries and framework

The libraries and frameworks were employed in this work mostly python libraries that used for ANNs, working with arrays and visualisation are listed below:

- **Tensorflow (2.5.0)** Tensorflow (Abadi et al. 2015) is a Python framework that provides an open source platform for ANNs. It has various tools, libraries and community resources that helps in building DL models. In this study, We trained the DL models using Tensorflow.
- **Keras (v2.5.0)** Keras (Chollet et al. 2015) is a open source DL library in python, acting like an interface for the TensorFlow framework.
- **NumPy (v1.19.5)** NumPy stands for "Numerical Python" which is one of the popular Python libraries that makes working with arrays easier. In this library, there are some functions for linear algebra, Fourier transform, and matrices calculation which was built by Travis Oliphant in 2005. Since it is an open-source project, it can be used free.
- **Matplotlib (v3.2.2)** Visualisation helps us to access a huge amount of data in digestible visual form. One of the popular and remarkable visualisation python libraries is Matplotlib. it is created based on Numpy which was introduced by John Hunter in 2002 (Barrett et al. 2005). There are several plots including line, bar, scatter, histogram etc in this library.

In the Github repository, users are required to clone the repository to use the modules and run notebooks. We strongly recommend deploying files following the folder structure of all README files. There are several directories in the repository including Gridsearch, Notebooks, animation, docs, images, src, saved-models and tables.

The source codes for the model can be found in the src directory including blocks.py, unet-uae.py and unet-att.py which are the python codes for blocks and layers. Documentation files can be found in the doc directory, which is generated to .html files with Sphenix. Experimental results can be found in the results directory. The result visualization is placed in the images and animation directories and the notebooks used to train models and compare and evaluated them can be found in Notebooks directory. The notebooks for grid-search and hyperparameter tuning saved in Gridsearch directory. The links of saved models are given in Readme and saved-models directory. The code metadata is summarized in table 1.

Table 1: code metadata used in this study

Code metadata description	Information
link to repository for this code	https://github.com/acse-srm3018/DeeplearningProxy
legal code licence	Apache-2.0 licence
software code language	Python
compilation requirement and dependencies	Windows, keras, python, numpy, matplotlib,Tensorflow

3 Code and Implementation

The model architectures are similar to the ones which are defined in previous sections. We trained four models including RR U-Net and A RR U-Net for pressure and saturation separately. Although 2.1 million parameters in our models to learn for all four models, the models do not suffer from overfitting. Zhu et al (Zhu & Zabaras 2018) suggested in their study that high dimensional data acts as a regulariser for our model and helps in decreasing overfitting problems.

In this section, more details about the dataset, the preprocessing of data, Loss functions, hyper-parameter tuning and other steps in training the models were presented.

3.1 Dataset

In this study, synthetic dataset were generated using Intersect (an industrial software). It consists of 3000 geological maps (permeability) and their corresponding pressure and saturation map. Most of datasets used for creating surrogate model of reservoir simulation are operated under well controls condition (Tang et al. 2020, Jin et al. 2019, Mo et al. 2019). However, in this study we used more variable geological models that make models to learn slower in training stage. Although the data was generated for 1800 months, the training data was collected at 10-time steps (between 3 months and 60 months). The dataset is then divided into 2250 training and 750 testing data that will be used for training ANN models. The network will be trained using the input and their corresponding pressure and saturation maps as the targets. The geological images are defined on 100*100 cells. The model contains 4 production wells and 1 injection well. For a particular channelized grid block, there are two channels, one with permeability equal to 10 md and the other with 1000 md.

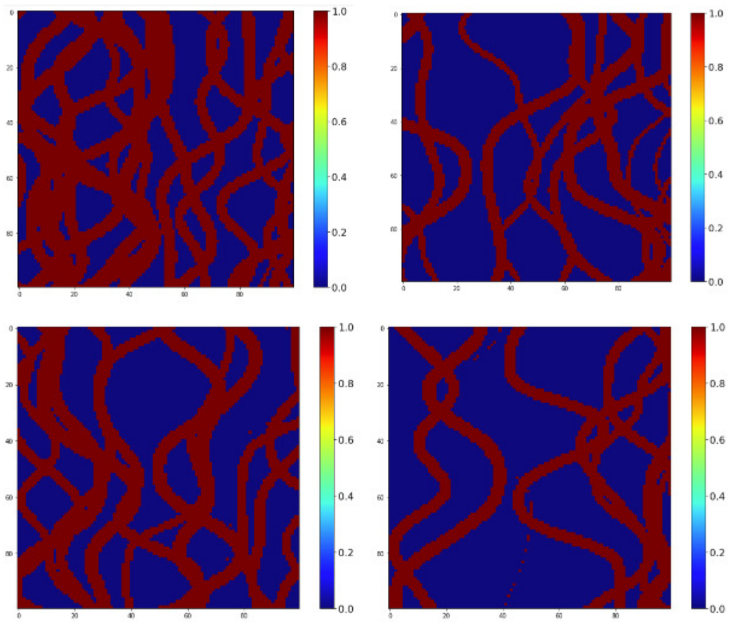


Figure 3: Four random permeability map from datasets

3.2 Data preprocessing

One of the vital steps before training a DL model is preprocessing because of the capability of the model to learn effects by data and its quality; therefore, we need to preprocess our data before it enters to model as input. Mean normalisation, standardization and rescaling are very common data preprocessing approaches. In Normalisation, we aim to do a conversion on data such that the mean of values becomes 0, the converted values then were divided by subtracting maximum and minimum of values. This helps in limiting the range of values to be close to zero and this will improve the training process. In this study, the input is permeability map with binary values of 0 and 1 for shale/mud and sand, respectively. The saturation values are between 0 and 1, while the pressure values have wide ranges. Therefore, They need to be normalised. Since in this study, we aim to predict the dynamic reservoir pressure, it needs to normalise the pressure valued on each time step based on their mean, maximum and minimum, separately. The normalised pressure formula which used in this study is equation1:

$$P^{i,t}_{normalised} = \frac{(P^{i,t} - P^{t}_{mean}) - min(P^{i,t})}{max(P^{i,t}) - min(P^{i,t})} \tag{1}$$

Where $P^{i,t}_{normalised}$ and $P^{i,t}$ are the normalised pressure and pressure at time step t and ith location in spatial map, respectively also $min(P^{i,t})$ and $max(P^{i,t})$ are the maximum and minimum of pressure at time step t. Some Numerical experiments were conducted and the results show this normalisation improves the model accuracy for pressure prediction.

3.3 Loss Function and Optimisation Algorithm

Loss functions is a function that used to optimize the model parameters. In this study, we used L1 and L2 loss functions (regularisation to avoid over-fitting) which are used to minimize the error for saturation and pressure, respectively. The difference between the value from simulator and the predicted value predicted from model defined as error. The squared error is considered the loss function L, and then adding L1 and L2 regularization terms to them. Our objective is to minimise the loss function. Different types of optimizers are used in DL based models. ADAM is one of the best optimization algorithms to update model parameters(weights and biases) based on training data.

3.4 Hyper-parameter Tuning

Hyper-parameters of DL based models such as batch size, type of optimizer, learning rate etc have a significant effect on the capability of model to learn; Therefore, it is indispensable to find the optimum values of hyper-parameters by grid searching among specified ranges of hyper-parameters. In this method, the DL model has been trained using the different combinations of parameters and find the optimum combination of them.

The batch size and learning rate are two hyper-parameters that are investigated in this study and the results for pressure models have been presented in the following tables. The first table presented the effect of changing batch size on loss. As it is evident from results, the batch size = 8 demonstrates the best performance.

Table 2: Effect of Batch Size

Hyper-parameters			A RR U-Net Loss		RR U-Net Loss	
Batch Size	Learning Rate	Optimizer	Training	Validation	Training	Validation
4	1e-4	Adam	0.61	1.81	0.84	2.31
8	1e-4	Adam	0.42	1.49	0.68	2.46
16	1e-4	Adam	0.56	1.78	1.29	3.72

The second table presents the effect of learning rate. The learning rate equals to 1e-4 for both models have the lowest error.

Table 3: effect of Learning rate

Hyper-parameters			A RR U-Net Loss		RR U-Net Loss	
Batch Size	Learning Rate	Optimizer	Training	Validation	Training	Validation
8	1e-2	Adam	6.84	8.14	13.23	32.54
8	3e-3	Adam	1.64	2.61	2.82	7.19
8	1e-4	Adam	0.42	1.49	0.69	4.06

3.5 Required time for training

The training time mostly depends on the hyper-parameters like batch size, learning rate, number of epochs, etc. The average time for training every four models with 2250 training datasets including 100*100 grids and 10-time steps and 300 epochs running on google colab GPU was about 300 minutes. The training elapsed time will be increased by rising the training data samples, the number of time steps and changing hyper-parameters. Fig. 7 represents how increasing the number of time steps effects on average run time per epochs. As it is obvious from graphs, the elapsed time does not linearly change with changing number of time steps.

Since our ultimate objective in this study is training a deep learning proxy model, using as reservoir simulator to reduce the simulation time in history matching. It still seems the elapsed training time still will be lower than simulation time using other simulators. However, it still needs to quantify in future studies.

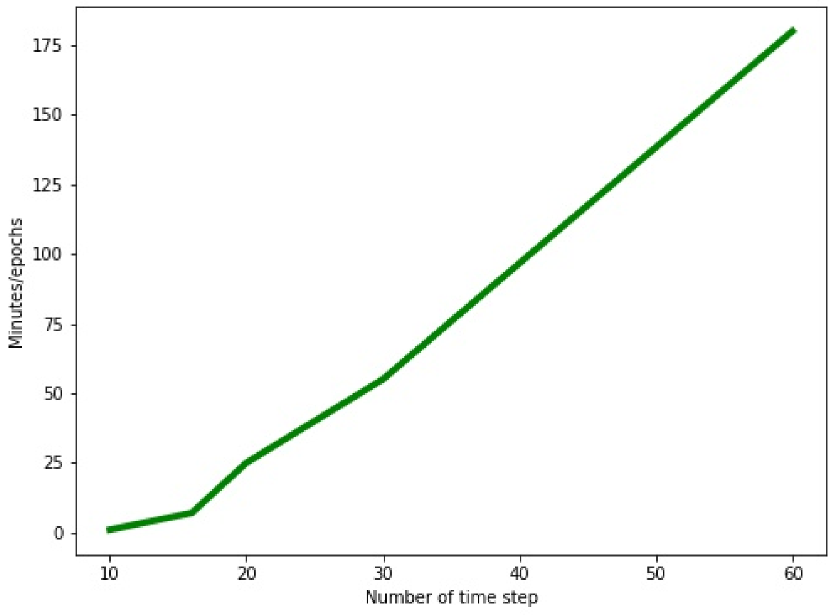


Figure 4: Elapsed time (minutes) per epochs for training versus dataset time steps

4 Results

In this section, the results are presented and discussed. The results are presented in two parts. First, the pressure and saturation maps predicted by the two models are presented and second, the relative error is discussed.

4.1 Evaluating the models by Reconstruction of pressure and saturation map from DL models

From a total 3000 generated data, 750 data was used to evaluate the model performance. We trained the model using 10 time steps from a total 61 time steps which we generated from Intersect software. Here , we presented the results of 4 time steps (including 90 days, 540 days, 1080 days, 1800 days) as representatives of results. The pressure and saturation for one model(channelised permeability map)

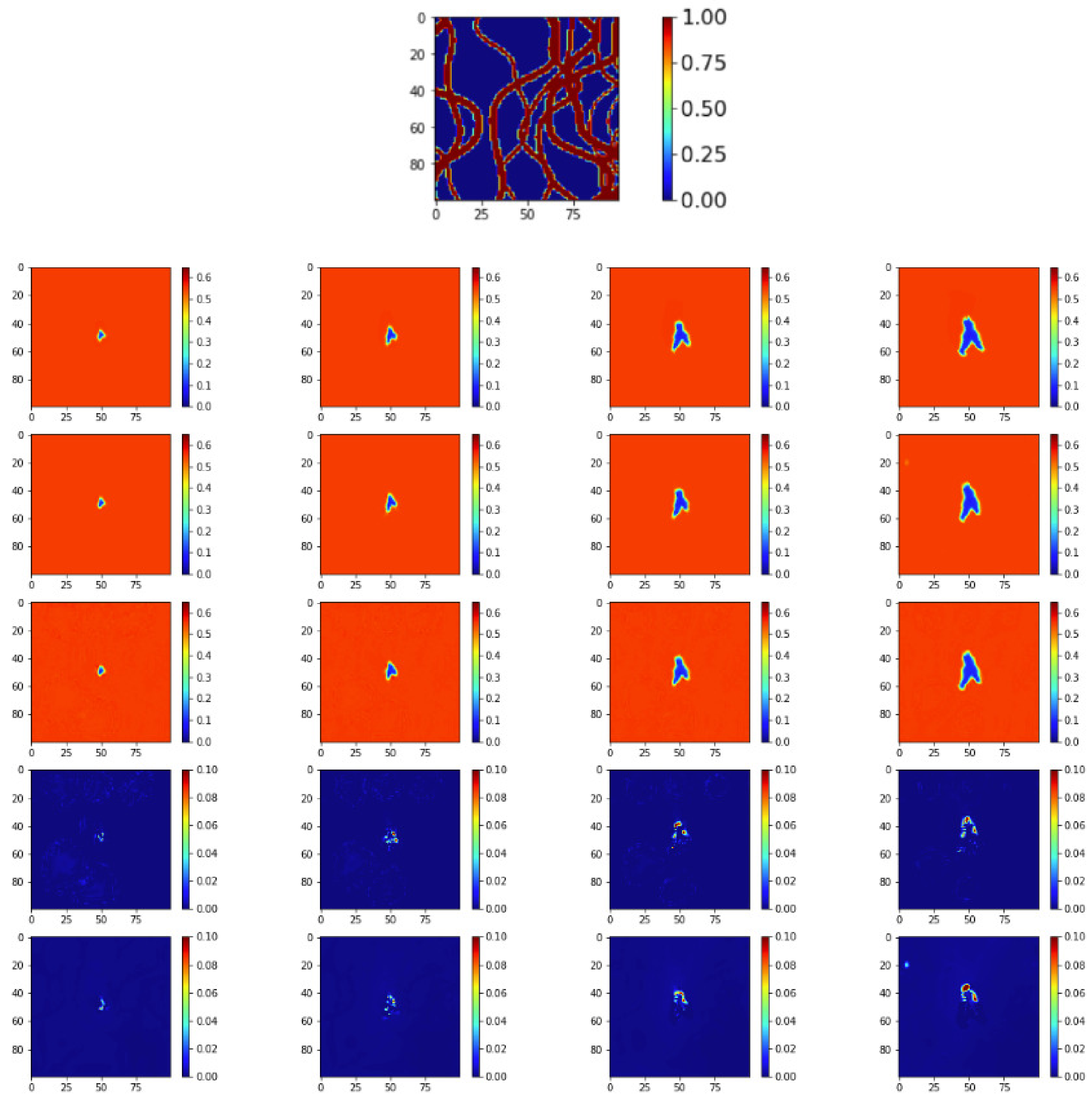


Figure 5: permeability map and its related saturation map from simulator and prediction from A RR-U-net and RR U-net models

have been shown in figure 4 and figure 5, respectively. The first row of these two figures show the permeability maps, the second row is the saturation and pressure maps from the simulator(Intersect), the third and 4th rows are the results from RR U-Net and A RR U-net, the 5th and 6th rows are the difference map between prediction from RR U-Net model and the state map from the simulator and A RR U-net model and state map from the simulator, respectively.

As it is obvious in Fig. 4 the sequence of saturation maps with time is dependent on the shape and position of channels in the permeability map, where the transportation mostly occurs along the channels with high permeability. It can be seen that both RR U-net and A RR U-Net models provide precise predictions for four-time steps. However, the results from A RR U-net show better accuracy in predictions with small errors noticeable near saturation fronts while predictions from RR U-Net error mostly occur in the channel.

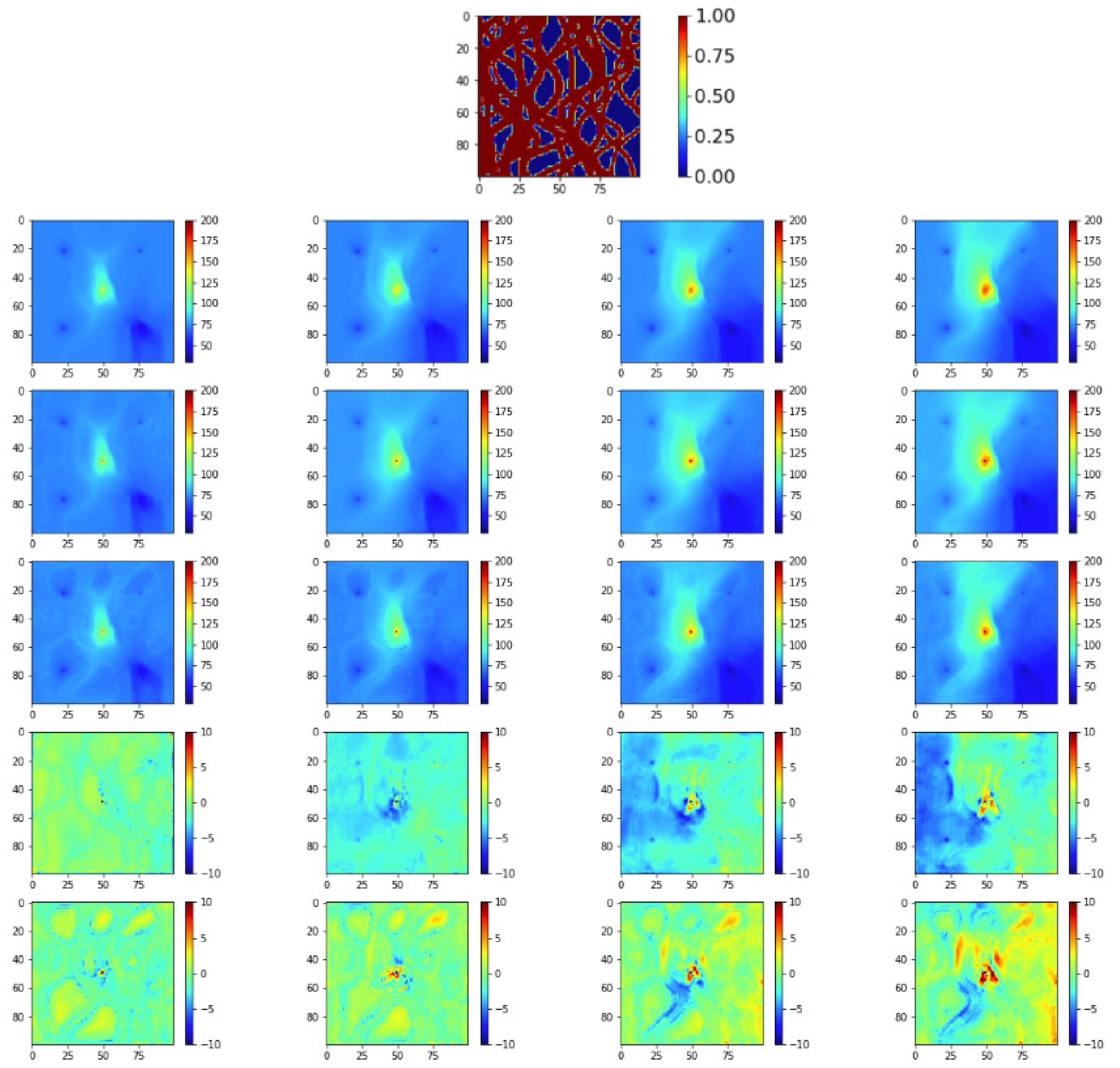


Figure 6: permeability map and its related pressure map from simulator and prediction from A RR-U-net and RR U-net models

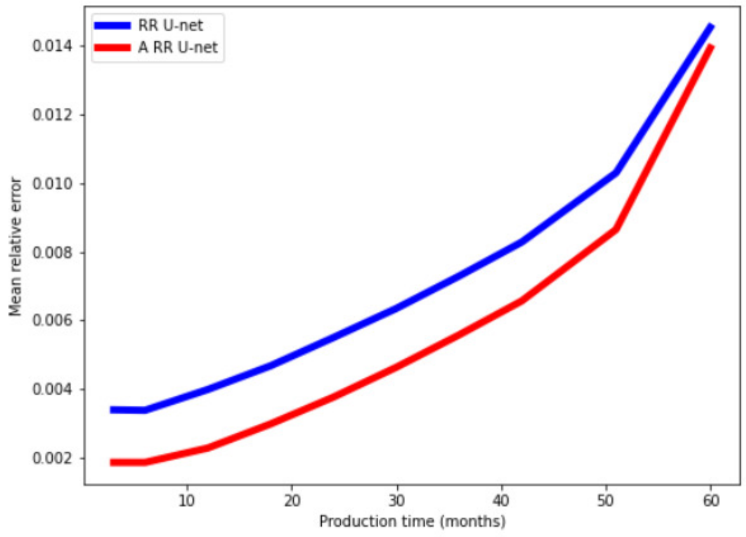
The pressure results in fig 5 indicate the high accuracy for both proxy models similar to what we have seen for saturation models. Although the accuracy is high for both models, it can be seen from the difference map between model and simulator that A RR-U-net is in comparison with another proxy model, i.e RR U-net (Tang et al. 2020).

4.2 Relative error

For further model evaluation, the relative error for pressure and saturation models (RR U-net and A RR U-Net) are quantified. For saturation, the mean relative error for all test datasets with 750 samples, size of grid blocks which is 100*100 and 10-time steps are calculated by the following equation:

$$E_{saturation} = \frac{1}{n_s n_g n_{ts}} \sum_{i=1}^{n_s} \sum_{j=1}^{n_g} \sum_{t=1}^{n_{ts}} \frac{||(S_{i,j,t}^{pred} - S_{i,j,t}^{sim})||}{S_{i,j,t}^{sim}} \tag{2}$$

Where n_s, n_g and n_{ts} are the number of samples, grids($100*100 = 10000$) and time steps, respectively. $S_{i,j,t}^{pred}$ and $S_{i,j,t}^{sim}$ are the saturation values from simulator and prediction of models at positions i, j and time step t. For 750 test data, the mean relative errors are 0.9% and 1.1% for the A RR U-net model and RR U-net model, respectively. Figure 6 has shown the mean relative error for saturation and pressure models over 750 test datasets at different time steps. It is evident from the figure that the relative error for the RR U-net model is constantly higher than the A RR U-net model at any time step.



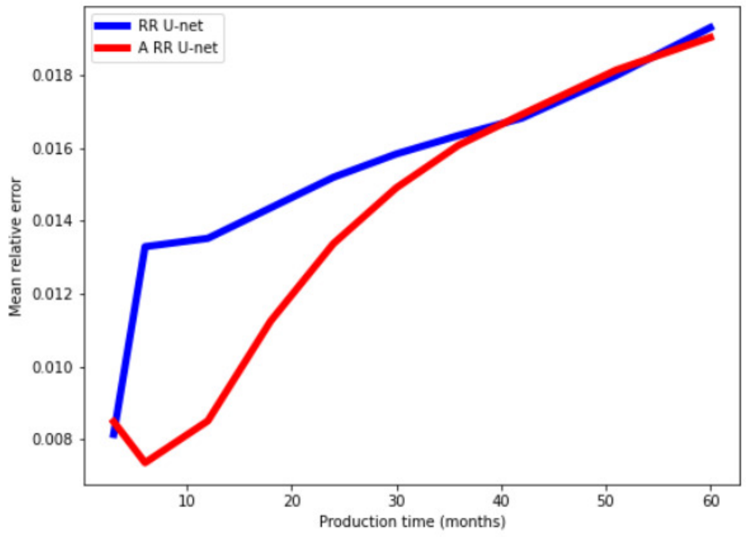


Figure 7: Saturation and pressure relative error versus production time: first row: Saturation, second row: Pressure

Similar to saturation, pressure relative error for 750 test samples, size of grid blocks which is 100* 100 and 10-time steps are calculated by:

$$E_{pressure} = \frac{1}{n_s n_g n_{ts}} \sum_{i=1}^{n_s} \sum_{j=1}^{n_g} \sum_{t=1}^{n_{ts}} \frac{||(P_{i,j,t}^{pred} - P_{i,j,t}^{sim})||}{P_{i,j,t}^{sim}} \tag{3}$$

Where n_s, n_g and n_{ts} are the number of samples, grids(100*100 = 10000) and time steps, respectively. $P_{i,j,t}^{pred}$ and $P_{i,j,t}^{sim}$ are the pressure values from simulator and prediction of models at positions i, j and time step t.

For 750 test data, the mean relative errors are 1.4% and 1.6% for the A RR U-net model and RR U-net model, respectively. Figure 6 has shown the mean relative error for pressure models over 750 test datasets in different time steps. It is evident from the figure that the relative error for the RR U-net model is higher than the A RR U-net model at early time steps but they shows similar results at later time steps.

Although for both saturation and pressure, our proposed A RR U-net indicates better results but still, there is an issue. The issue is that the mean relative error increase by increasing the time steps. one possible solution is to run the model for longer time steps to see if relative error reaches stability. We tried to run it for more time steps, but since it needs lots of memory and CPU to run, we fail to complete the tasks. This can be addressed in future works.

5 Conclusion

Two DL based models were trained to use as a proxy for simulating 2D reservoir pressure and saturation with time. These two models included CNN blocks together with LSTM. RR U-net and A RR U-net are used as two DL architectures. The synthetic data are saturation and pressure maps simulated with the reservoir software Intersect at different time steps and from 3000 channelised permeability maps. Then, the DL models were trained using 2250 cases of this dataset for 10 time steps.

The performance of trained models evaluates using a 750 test dataset. The relative error was calculated for both saturation and pressure models. By comparison between two models results, it can be demonstrated that although both models can forecast the results accurately, A RR U-Net shows higher accuracy in forecasting the pressure and saturation map in comparison with results from the simulator(Intersect). This is because attention blocks helps network to learn from more relevant and important information by giving each of them specific weights. The A RR U-Net model also takes advantages of residual layers as well. This layers assist network to improve the accuracy by building a shorter path (skip connections) between layers near input and output.

Several factors such as batch size, number of epochs, etc influence the required time for training. In the training stage, about 1 minute takes for each epoch to run for the dataset including 2250 training data and 10-time steps.

The elapsed time for training increases by increasing the number of time steps. However, increasing the elapsed time is not linear. For example, the average time takes the network to train are 25 minutes/epoch, 55 minutes/epochs for 20-time step and 30-time step, respectively. Besides, as the number of time steps rises, the complexity of the models will increase. Therefore, it needs more epochs to reach stable and reasonable training and validation loss and relative error.

The first results presented in this thesis are calling for improvements and further works and three points are now discussed.

- The ultimate objective of this work is to speed up the history matching by using the DL based model instead of high-fidelity numerical simulation. Therefore, in future works, the models can be applied for history matching and the speed compared with the simulator one. This can be done by monitoring elapsed time for calculating flow rate using simulators and models.

- The DL models in this study trained for a sequence of 10-time steps (for 5 years production every 6 months). It is worth training models for wider time steps (20 or 30-time steps) and longer production time (more than 5 years). In this study, we used google colab GPU for training models. Since training models for longer time steps may take several days using GPU. As a result of being computational-expensive, the Training model needs higher memory and GPU sources for longer production time and more time steps.
- A RR U-Net is suggested to use as proxy models in this study. Its results are more accurate in comparison to RR U-Net. Since it is applied to 2D reservoir models, it can be applied to 3D reservoir data in future studies to make a comparison between performances of A RR U-Net and RR U-Net.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. & Zheng, X. (2015), 'TensorFlow: Large-scale machine learning on heterogeneous systems'. Software available from tensorflow.org.
URL: *https://www.tensorflow.org/*

Alom, M. Z., Yakopcic, C., Hasan, M., Taha, T. M. & Asari, V. K. (2019), 'Recurrent residual u-net for medical image segmentation', *Journal of Medical Imaging* **6**(1), 014006.

Alpak, F. O. et al. (2010), 'A mimetic finite volume discretization method for reservoir simulation', *SPE Journal* **15**(02), 436–453.

Barrett, P., Hunter, J., Miller, J. T., Hsu, J.-C. & Greenfield, P. (2005), matplotlib–a portable python plotting package, *in* 'Astronomical data analysis software and systems XIV', Vol. 347, p. 91.

Cheung, S. W., Chung, E. T., Efendiev, Y., Gildin, E., Wang, Y. & Zhang, J. (2020), 'Deep global model reduction learning in porous media flow simulation', *Computational Geosciences* **24**(1), 261–274.

Chollet, F. et al. (2015), 'Keras'.
URL: *https://github.com/fchollet/keras*

Ertekin, T., Abou-Kassem, J. H. & King, G. R. (2001), *Basic applied reservoir simulation*, Vol. 7, Society of Petroleum Engineers Richardson, TX.

Fung, L.-K., Hiebert, A., Nghiem, L. X. et al. (1992), 'Reservoir simulation with a control-volume finite-element method', *SPE Reservoir Engineering* **7**(03), 349–357.

Goldberg, Y. (2016), 'A primer on neural network models for natural language processing', *Journal of Artificial Intelligence Research* **57**, 345–420.

Graves, A., Mohamed, A.-r. & Hinton, G. (2013), Speech recognition with deep recurrent neural networks, *in* '2013 IEEE international conference on acoustics, speech and signal processing', Ieee, pp. 6645–6649.

Jin, K. H., McCann, M. T., Froustey, E. & Unser, M. (2017), 'Deep convolutional neural network for inverse problems in imaging', *IEEE Transactions on Image Processing* **26**(9), 4509–4522.

Jin, Z. L., Liu, Y. & Durlofsky, L. J. (2019), 'Deep-learning-based reduced-order modeling for subsurface flow simulation', *arXiv preprint arXiv:1906.03729* .

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

Karumuri, S., Tripathy, R., Bilionis, I. & Panchal, J. (2020), 'Simulator-free solution of high-dimensional stochastic elliptic partial differential equations using deep neural networks', *Journal of Computational Physics* **404**, 109120.

Liu, Y., Sun, W. & Durlofsky, L. J. (2019), 'A deep-learning-based geological parameterization for history matching complex models', *Mathematical Geosciences* **51**(6), 725–766.

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

Padmanabha, G. A. & Zabaras, N. (2021), 'Solving inverse problems using conditional invertible neural networks', *Journal of Computational Physics* **433**, 110194.

Ronneberger, O., Fischer, P. & Brox, T. (2015), U-net: Convolutional networks for biomedical image segmentation, *in* 'International Conference on Medical image computing and computer-assisted intervention', Springer, pp. 234–241.

Sainath, T. N., Vinyals, O., Senior, A. & Sak, H. (2015), Convolutional, long short-term memory, fully connected deep neural networks, *in* '2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)', IEEE, pp. 4580–4584.

Swapna, G., Kp, S. & Vinayakumar, R. (2018), 'Automated detection of diabetes using cnn and cnn-lstm network and heart rate signals', *Procedia computer science* **132**, 1253–1262.

Tang, M., Liu, Y. & Durlofsky, L. J. (2020), 'A deep-learning-based surrogate model for data assimilation in dynamic subsurface flow problems', *Journal of Computational Physics* **413**, 109456.

Tang, M., Liu, Y. & Durlofsky, L. J. (2021), 'Deep-learning-based surrogate flow modeling and geological parameterization for data assimilation in 3d subsurface flow', *Computer Methods in Applied Mechanics and Engineering* **376**, 113636.

Vo, H. X. & Durlofsky, L. J. (2015), 'Data assimilation and uncertainty assessment for complex geological models using a new pca-based parameterization', *Computational Geosciences* **19**(4), 747–767.

Wang, Y. & Lin, G. (2020), 'Efficient deep learning techniques for multiphase flow simulation in heterogeneous porousc media', *Journal of Computational Physics* **401**, 108968.

Yoo, Y., Yun, S., Jin Chang, H., Demiris, Y. & Young Choi, J. (2017), Variational autoencoded regression: high dimensional regression of visual data on complex manifold, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition', pp. 3674–3683.

Zhao, X., Wu, Y., Song, G., Li, Z., Zhang, Y. & Fan, Y. (2018), 'A deep learning model integrating fcnns and crfs for brain tumor segmentation', *Medical image analysis* **43**, 98–111.

Zhou, Y. (2012), Parallel general-purpose reservoir simulation with coupled reservoir models and multisegment wells, PhD thesis, Stanford University.

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