



Cumulative oil production in flow unit-crossing wells estimated by multilayer perceptron networks

Edvaldo F. M. Neto¹ · Gustavo P. Oliveira¹ · Rafael M. Magalhães¹ · Leonardo V. Batista² · Lucídio A. F. Cabral³ · Moisés D. Santos¹

Received: 26 December 2020 / Accepted: 16 April 2021 / Published online: 1 May 2021
© The Author(s) 2021

Abstract

Knowing the ultimate oil production in wells is a crucial point for reservoir planning and management to anticipate value for money. Commercial reservoir simulators are able to predict production curves with high confidence, but repetitive tasks in a few cases may spend a precious time of staff as well as require a large computational effort. Although artificial intelligence (AI) is providing an alternative path to the usual workflow, many commercial simulators lack robust AI algorithms. This work introduces a methodology based on a multilayer perceptron (MLP) neural network to predict the final cumulative oil production of a reservoir at vertical wells that cross hydraulic flow units (HFUs), which are volumes endowed with good flow attributes. Each well location is attached to special spots previously determined from clustering and calculation of maximum closeness centrality points (MaxCs) within a class of HFUs. The database is divided into training, validation, and testing sets organized after processing the UNISIM-I-D synthetic model, representative of the Namorado Field, Campos Basin, Brazil. The key rationale of this paper is to use the feature of MaxCs of being drivers for well placement as knowledge base to learn the production mechanisms of the oilfield. The outcomes are presented from two perspectives: an original MLP and its post-processed version. Both are compared with reservoir simulations carried out in CMG Imex® and achieve reasonable agreement. The performance is measured by root-mean-squared error (RMSE) and mean absolute scaled error (MASE) both in original and post-processed versions. We show that average RMSE and MASE values near 0.07 and 14.00, respectively, are achieved without post-processing. With post-processing, gains of up to 43% are reported for the integral oil volume.

Keywords Petroleum reservoirs · Artificial intelligence · Oil recovery · Hydraulic flow units

Introduction

The accurate prediction of recoverable oil volume in petroleum reservoirs is a leading goal of the industry which demands time, financial resources, cutting-edge

technology, and a variety of operational services (Chakra et al. 2013). When foreseeing the in-place productive capacity of reservoirs, engineers can effectively think on sustainable projects, eschew ill-advised decisions and avoid short-sighted investments (Liu et al. 2020).

✉ Gustavo P. Oliveira
gustavo.oliveira@ci.ufpb.br

Edvaldo F. M. Neto
emeloppgi@gmail.com

Rafael M. Magalhães
rafael@dcx.ufpb.br

Leonardo V. Batista
leonardo@ci.ufpb.br

Lucídio A. F. Cabral
lucidio@ci.ufpb.br

Moisés D. Santos
mdantas@ci.ufpb.br

¹ Petroleum Engineering Modeling Laboratory, Federal University of Paraíba, Center of Informatics - João Pessoa Campus, João Pessoa, Brazil

² Computer Vision Laboratory, Federal University of Paraíba, Center of Informatics - João Pessoa Campus, João Pessoa, Brazil

³ Postgraduate Program in Informatics, Federal University of Paraíba, Center of Informatics - João Pessoa Campus, João Pessoa, Brazil

Nonlinearity and inhomogeneity, however, are inherent characteristics of rock and reservoir fluids that hinder any attempts of estimating production incorrigibly (Aizenberg et al. 2016).

Numerical simulations are considered one of the most effective methods to reproduce the physical behavior of reservoirs whenever pressure, volumetric rates or other quantity dynamically varying in space and time are desired for production estimation (Mamudu et al. 2020). However, they can be time-consuming and require massive computational effort. In contrast, artificial neural networks (ANNs) try to take the lead in efficiency by using a learning-based approach. ANNs are parallel distributed systems capable to represent complex and nonlinear relationships through input and output patterns from experimental data (Berneti and Shahbazian 2011). They overcome all the conventional methods based on statistics, such as autoregressive integrated moving average (ARIMA), autoregressive moving average (ARMA), and autoregressive conditioned heteroskedasticity (Chakra et al. 2013), which turns it into a potential choice to learn about non-deterministic aspects in physical processes occurring in reservoirs.

Several papers encompassing the application of neural networks for oil recovery prediction were published by implementing singular or combined approaches, such as the productivity potential (PP), used as input data to estimate cumulative oil production and establish optimal perforations with reducing the three-dimensional distribution of properties to a two-dimensional map (Min et al. 2011). Applications observed in data of real oilfields are plentiful. A model known as higher-order neural network (HONN) was developed to estimate the oil production in the Cambay Basin, India (Chakra et al. 2013). A derivative-free multilayer network with multivalued neurons and complex weights was used to predict the oil recovery at an oilfield in the Gulf of Mexico Aizenberg et al. (2016). More recently, networks based on fuzzy clustering, genetic algorithm, and memory-like were employed to predict the oil production in China's oilfields (Hu et al. 2018; Sagheer and Kotb 2019; Liu et al. 2020). With a hybrid system formed by ANNs and Bayesian networks, Mamudu et al. (2020) have proposed to filling the gap of simulators concerning the relationship between oil recovery and associated risks.

Despite that, with exception of Min et al. (2011), all the references aforementioned explored the usage of neural networks to estimate the oil recovery of single wells by using history data bonded to each well per se. While this perspective is appropriate to gain insight on the future behavior of a given producer well—even though the limitation of well information hampers the full unleash of data training power—attempts toward the generalization of this kind of learning when considering a scenario of multiple wells are unknown at the best of our knowledge.

Wells are usually modeled as a set of discrete cells in corner-point grids. Since these synthetic reservoir models are usually built to have thousands of cells with specified properties, unpack them into arrays that will serve as input data to an ANN may be impractical. A way to minimize this cost in terms of learning is to pick up smaller pieces of information and form groups of interest. In reservoir modelling, rock clusters whose petrophysical and geological properties are similar can be recognized as a hydraulic flow unit (HFU). HFUs are volumes of a reservoir with favoured ability to convey subsurface fluids. This way, flow unit models enable us to select “chunks” of geologic information around wells so that special cells—so-called maximum closeness centrality cells (MaxCs)—known a priori as flow convergence zones are used to construct input arrays with reduced dimension and high quality to describe the production mechanisms necessary to feed the network learning (Oliveira et al. 2016, 2020).

In this paper, we implemented MLP architectures to predict the cumulative oil production in petroleum reservoirs coupled with the HFU model and MaxCs. The key rationale is to use the feature of MaxCs of being drivers for well placement as knowledge base to learn the production mechanisms of the oilfield. This application is grounded on the case study UNISIM-I-D, a model for the Namorado Oilfield, Campos Basin, Brazil Avansi and Schiozer (2015). We detected 54 MaxCs and selected each vertical column of cells associated to them as a well model, thus forming the main dataset. Next, we prepared the dataset by separating 44 wells for training, 5 for validation, and 5 for testing. The outcomes are presented from two perspectives: an original MLP and its post-processed version. Both are compared with reservoir simulations carried out in CMG Imex[®] and achieve reasonable agreement. The performance is measured by root-mean-squared error (RMSE) and mean absolute scaled error (MASE) both in original and post-processed versions. With post-processing, gains of up to 43% are reported for the integral oil volume. We show that both versions of the MLP (original and post-processed) and numerical simulation outcomes agree in different metrics reaching averages of 0.0724 in RMSE and 14.2791 in MASE for the 5 testing wells.

Background

Hydraulic flow units

Hydraulic flow units (HFU) are regions in a reservoir that have similar characteristics and privileged flow conditions. Identifying them is useful to better characterize the reservoir and understand the local relationship between porosity and permeability. One of the well-known methods to identify HFUs is reservoir quality index (RQI) / flow zone indicator



(FZI) (Amaefule et al. 1993), which writes the Kozeny–Carman equation as

$$\sqrt{\frac{k}{\phi_e}} = \frac{1}{\sqrt{F_s} \tau S_{V_{gr}}} \frac{\phi_e}{1 - \phi_e}, \quad (1)$$

where k is the absolute permeability, ϕ_e is the effective porosity, F_s is the grain shape factor, τ is pore network tortuosity, and $S_{V_{gr}}$ is the surface area per unit grain volume. By defining

$$RQI = 0.0314 \sqrt{\frac{k}{\phi_e}} \quad \text{and} \quad FZI = \frac{1}{\sqrt{F_s} \tau S_{V_{gr}}}, \quad (2)$$

where 0.0314 is a conversion factor to millidarcies, Eq. (1) can be rewritten as

$$RQI = FZI \times \phi_z. \quad (3)$$

Here, $\phi_z = \frac{\phi_e}{1 - \phi_e}$ is the pore-to-matrix ratio. With applying the log function on both sides of Eq. 3,

$$\log(RQI) = \log(FZI) + \log(\phi_z), \quad (4)$$

we can define a flow unit as follows: in the log-log plot of RQI versus ϕ_z , all samples with similar FZI values will be located around a straight line with a unitary slope, meaning that they correspond to a rock sample with similar attributes and, therefore, constitute a HFU.

Since FZI is a continuous variable, it is common to convert it to discrete rock types (DRT) as proposed by Guo et al. (2005) by using

$$DRT = \text{round}[2\ln(FZI) + 10.6], \quad (5)$$

where round returns an integer distribution. Each DRT then tags a rock type according to the reservoir's heterogeneity.

Flow unit clustering

To obtain different clusters that will form HFUs, we group them according to the DRT values of each cell in the reservoir through logical masks (Oliveira et al. 2016). For example, Figure 1 illustrates two cluster models.

Oliveira et al. (2016) have examined the influence of HFU connectivity in the choice of perforation strategies to improve oil recovery. To establish them, metrics were used to classify vertices and identify special roles played by them in relation to other vertices of the volume. Among such metrics, closeness centrality was able to generate perforations with competitive recovery factors. Closeness centrality γ computes how close a given cell is from all others in the volume through cell-to-graph mapping. Its formula reads as

$$\gamma(v_q) = \frac{1}{\sum_{i=1}^{n_q} d(v_q, v_q^i)}, \quad (6)$$

where n_q is the number of cells of the cluster and $d(v_q, v_q^i)$ is the shortest distance from v_q to v_q^i . From Eq. (6), a single maximum closeness centrality cell (MaxC) can be ascribed to each HFU, thereby determining a good candidate for well completion. Figure 2 illustrates the application of γ to the three-dimensional clusters appearing in Fig. 1.

Artificial neural networks

Artificial neural networks (ANNs) are mathematical models inspired in biological structure of neurons. The simplest model employed in scientific problems is the McCulluch–Pitts' multi-layer perceptron (MLP). A MLP network defines a mapping $\mathbf{y} = f(\mathbf{x}; \mathbf{w})$ of input values \mathbf{x} onto output values through learning of values for the array of parameters \mathbf{w} that lead to the best approximation of the function

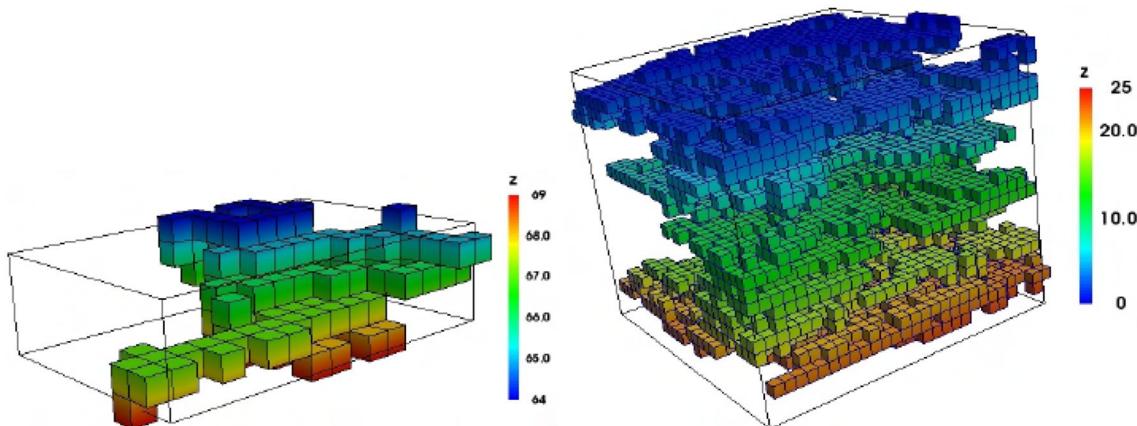


Fig. 1 Examples of clustered cells of the reservoir model forming two different hydraulic flow units: one related to DRT = 13 with 111 cells (left side); another related to DRT = 14 with 4989 cells (right side).

The color axis z here indicates the depth layers that bound the clusters. Adapted from (Oliveira et al. 2016)

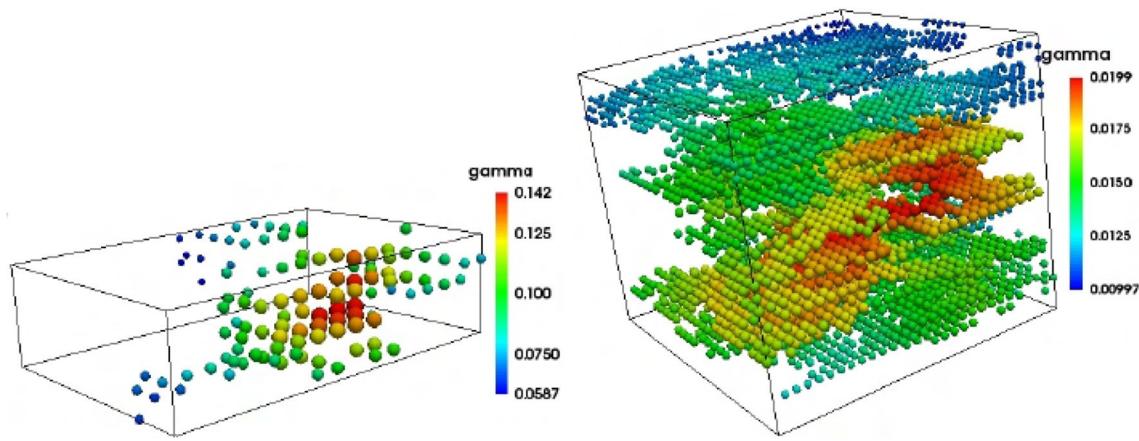


Fig. 2 3D point representation of the closeness centrality scattered over two clusters (flow unit models): DRT = 13 on the left side and DRT = 14 on the right side. The cell whose closeness centrality is maximum (MaxC) is placed amidst the reddish region, where the

color tone is more intense. MaxCs are used as reference points for associated production columns (producer wells) which, later on, will form the dataset for network training. Adapted from (Oliveira et al. 2016)

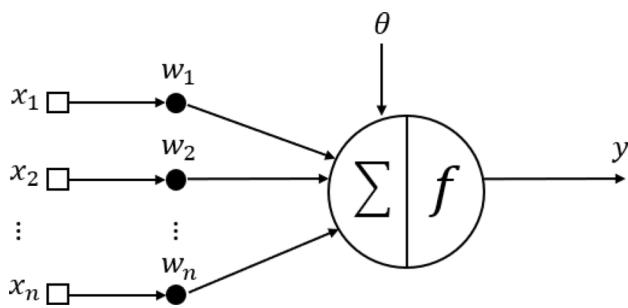


Fig. 3 McCulloch–Pitts' model of an artificial neuron

f (Goodfellow 2016). MLPs require little computational effort and can generalize the information learned from training examples.

An artificial neuron, as that illustrated in Fig. 3, is composed by n input terminals x_1, x_2, \dots, x_n , que generate a single output y .

The input data are balanced through n weights w_1, w_2, \dots, w_n which grade the importance of each input in the calculation. Once the products $x_i w_i$ are computed, they are biased by a quantity θ so that the local field

$$\mu(\mathbf{x}) = \sum_{i=1}^n x_i w_i + \theta = \mathbf{x}^\top \mathbf{w} + \theta, \quad (7)$$

is a linear model that will be sent to the activation function f . Usually, the activation function is such that

$$y = f(\mu) \quad (8)$$

whose formation law is nonlinear. Whether the μ surpasses a given threshold, f returns a state of active neuron.

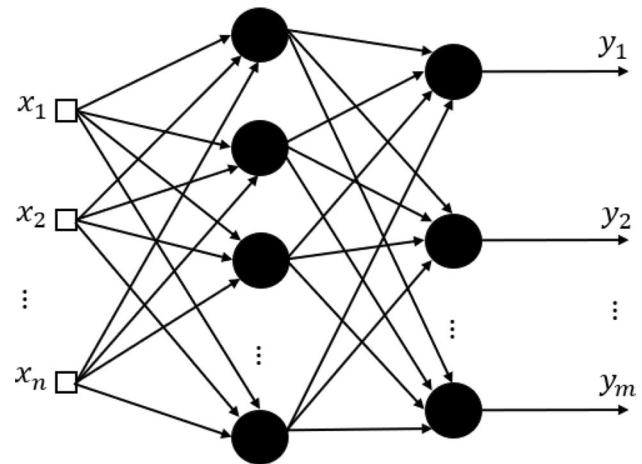


Fig. 4 Architecture of a multi-layer perceptron network with n input values (input layer), m output values (output layer) and two intermediary layers (hidden layers) with several neurons

ANNs can solve highly complex problems. A MLP is featured by connected neurons arranged in many layers that form an architecture. The leftmost layer is called the input layer, the rightmost layer is called the output layer, and the intermediary layers are known as hidden layers, where most of the computational effort is allocated. Figure 4 illustrates a MLP with n input and m output values. As the number of hidden layers grows, the process overcomes the “shallow” learning and tends to what is coined as deep learning.

The learning of a MLP consists of an iterative fitting process whose objective is to find the vector \mathbf{w} of weights that, together, will minimize the error between the desired output and the output predicted by the MLP. The error expression is defined by the convenient form



$$E = \frac{1}{2} \|\mathbf{y} - \tilde{\mathbf{y}}\|_2^2 \quad (9)$$

where \mathbf{y} is the desired output, $\tilde{\mathbf{y}}$ is the predicted output, and $\|\cdot\|_2$ is the L_2 -norm. The most popular training algorithm for this network is known as backpropagation, which aims to propagate the error obtained in the output layer backward until getting to the first hidden layer.

Because the MLP learning occurs after training, the dataset available should admit subsets of validation and testing. While the validation subset evaluates the performance of the neural network during the training step, the testing set is placed apart and checks the ability of the network to interpret unknown information. Such process is commonly called generalization, since the network is not aware of the testing set beforehand. The results obtained during validation steps allow us to evaluate a series of factors that affect the MLP performance, mainly the occurrence of underfitting or overfitting. Whenever its hyperparameters are adjusted, the network goes through a new training-validation cycle that is repeated until satisfactory responses are achieved.

Performance metrics

The quantitative measurement of a model's performance is made through metrics. A common metric is the root-mean-squared error (RMSE), which is a scale-dependent error useful to compare methods applied to the same data set but not suitable for comparing data sets with different scales (Hyndman and Koehler 2006). RMSE is defined as

$$\text{RMSE} = \frac{\|\mathbf{y} - \tilde{\mathbf{y}}\|_2^2}{n^{1/2}}, \quad (10)$$

n is the number of input data.

Hyndman and Koehler (2006) proposed the mean absolute scaled error (MASE), which is a symmetric measure that also penalizes large-scale or small-scale errors, either positive or negative. Additionally, it never results in infinite or indefinite values. It is written as

$$q_j = \frac{\mathbf{y}_j - \tilde{\mathbf{y}}_j}{\text{MAE}}, \quad j = 1, 2, \dots, n. \quad (11)$$

where $\text{MAE} = \frac{1}{n-1} \sum_{i=2}^n |\mathbf{y}_i - \mathbf{y}_{i-1}|$ is the mean absolute error, and q_j is the scale error. Therewith,

$$\text{MASE} = \text{mean}(|q_j|), \quad (12)$$

where mean is the function that computes the arithmetic mean.

Methods

Reservoir base model

The case study of this paper is based on the UNISIM-I-D synthetic reservoir model of the Namorado Oilfield, located in the Campos Basin, Brazil Avansi and Schiozer (2015). The stratigraphic grid has a resolution of $81 \times 58 \times 20$ with 36739 active cells. Figure 5 depicts the porosity field of the model.

Data pre-processing

Data preparation

FZI and DRT values were computed for each cell of the model and grouped consistently to form HFU volumes (see Subsection 2.1). Next, for each HFU, the closeness centrality (see Eq. (6)) was calculated to obtain the reference points (MaxCs) (Roque et al. 2017), thus yielding 54 wells of interest. Figure 6 shows how a producer well is mounted by crossing the HFU vertically over the column in which the MaxC (red color) lies. We underline that the HFU generally has an irregular structure. Although a few cells along the column may not belong to the cluster (unfilled ones), the perforations are enforced over the entire column.

Out of the 54 wells selected, 44 were used for training, 5 for validation, and 5 for testing of the neural network. The following reservoir properties were handled and prepared with the software CMG Builder® to work as input data: oil saturation, pressure, permeability, and porosity. Time steps were set to vary monthly from 2020 to 2040, thus yielding a total of 241 months as depicted in Fig. 7.

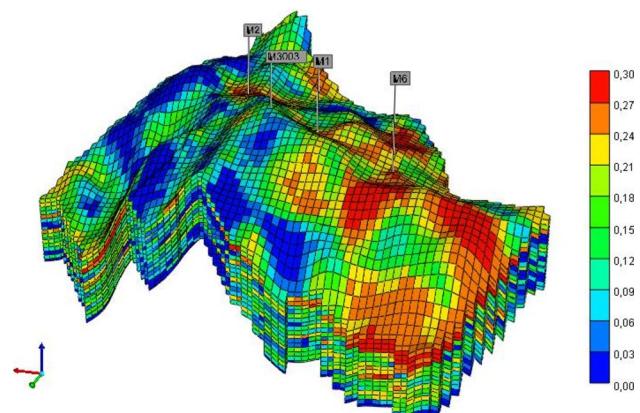


Fig. 5 3D view of the porosity field of the UNISIM-I-D synthetic model highlighting a few wells pre-chosen from the centrality metric. Post-processed from (Avansi and Schiozer 2015)

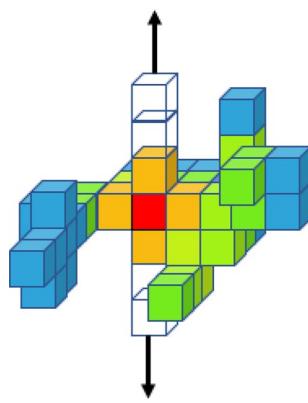


Fig. 6 Selection of a discrete well for study from an irregular cluster (HFU model). Colored cells represent the closeness centrality metric pointing out the MaxC (red color) as the reference cell to form the production column. Unfilled cells are neighbor cells up and down MaxC which are also perforated even though they do not belong to the cluster domain

Row entries represent time instants, whereas column entries identify the properties evaluated at the wells. Each well has a fixed length of 16 cells, what leads to 64 features per well. On the left side, we show the matrix corresponding to the first well. In the middle, we show the extended form of the input matrix which, in fact, contains all submatrices of each training well. On the right side, we show the condensed

form of the input matrix. Likewise, we can conclude that the validation and testing matrices are sized in 1205 x 64.

As a result, the MLP output vector—initially retrieved from the flow simulator—is an array with shape 10604 x 1 that stores the cumulative oil production estimated by each well along 241 months of simulation. In the same manner, one verifies that the validation and testing matrices are sized in 1205 x 1.

Data normalization

Data normalization accelerates the learning of the neural network and eases the pattern discovery amidst the data set. In this paper, we used the equation

$$\mathbf{x}^* = \frac{\mathbf{x} - \text{mean}(\mathbf{x})}{\text{std}(\mathbf{x})}, \quad (13)$$

to normalize the original data \mathbf{x} by using the arithmetic mean and the standard deviation.

Network setup

In this paper, we used a MLP network formed by 5 hidden layers composed individually with 51 neurons adjusted by dropout probability (Srivastava et al. 2014). The input vector is 64-dimensional, in accordance with the input matrix's

Fig. 7 Input matrix associated to the 44 training wells described under different perspectives. Row and column entries represent, respectively, time instants and well properties. Left side: submatrix relative to the first well (computer array with shape 241 x 64); middle: extended input matrix containing all submatrices; right: condensed form of the input matrix (shape 10604 x 64)

<table border="1" style="border-collapse: collapse; width: 100%; text-align: center;"> <tr><td>$P_{1,1}^1$</td><td>$P_{1,2}^1$</td><td>\cdots</td><td>$P_{1,64}^1$</td></tr> <tr><td>$P_{2,1}^1$</td><td>$P_{2,2}^1$</td><td>\cdots</td><td>$P_{2,64}^1$</td></tr> <tr><td>\vdots</td><td>\vdots</td><td>\ddots</td><td>\vdots</td></tr> <tr><td>$P_{241,1}^1$</td><td>$P_{241,2}^1$</td><td>\cdots</td><td>$P_{241,64}^1$</td></tr> </table> <table border="1" style="border-collapse: collapse; width: 100%; text-align: center;"> <tr><td>$P_{1,1}^1$</td><td>$P_{1,2}^1$</td><td>\cdots</td><td>$P_{1,64}^1$</td></tr> <tr><td>$P_{2,1}^1$</td><td>$P_{2,2}^1$</td><td>\cdots</td><td>$P_{2,64}^1$</td></tr> <tr><td>\vdots</td><td>\vdots</td><td>\ddots</td><td>\vdots</td></tr> <tr><td>$P_{241,1}^1$</td><td>$P_{241,2}^1$</td><td>\cdots</td><td>$P_{241,64}^1$</td></tr> </table> <table border="1" style="border-collapse: collapse; width: 100%; text-align: center;"> <tr><td>$P_{1,1}^2$</td><td>$P_{1,2}^2$</td><td>\cdots</td><td>$P_{1,64}^2$</td></tr> <tr><td>$P_{2,1}^2$</td><td>$P_{2,2}^2$</td><td>\cdots</td><td>$P_{2,64}^2$</td></tr> <tr><td>\vdots</td><td>\vdots</td><td>\ddots</td><td>\vdots</td></tr> <tr><td>$P_{241,1}^2$</td><td>$P_{241,2}^2$</td><td>\cdots</td><td>$P_{241,64}^2$</td></tr> </table> <table border="1" style="border-collapse: collapse; width: 100%; text-align: center;"> <tr><td>$P_{1,1}^3$</td><td>$P_{1,2}^3$</td><td>\cdots</td><td>$P_{1,64}^3$</td></tr> <tr><td>$P_{2,1}^3$</td><td>$P_{2,2}^3$</td><td>\cdots</td><td>$P_{2,64}^3$</td></tr> <tr><td>\vdots</td><td>\vdots</td><td>\ddots</td><td>\vdots</td></tr> <tr><td>$P_{241,1}^3$</td><td>$P_{241,2}^3$</td><td>\cdots</td><td>$P_{241,64}^3$</td></tr> </table> <table border="1" style="border-collapse: collapse; width: 100%; text-align: center;"> <tr><td>$P_{1,1}^{44}$</td><td>$P_{1,2}^{44}$</td><td>\cdots</td><td>$P_{1,64}^{44}$</td></tr> <tr><td>$P_{2,1}^{44}$</td><td>$P_{2,2}^{44}$</td><td>\cdots</td><td>$P_{2,64}^{44}$</td></tr> <tr><td>\vdots</td><td>\vdots</td><td>\ddots</td><td>\vdots</td></tr> <tr><td>$P_{241,1}^{44}$</td><td>$P_{241,2}^{44}$</td><td>\cdots</td><td>$P_{241,64}^{44}$</td></tr> </table>	$P_{1,1}^1$	$P_{1,2}^1$	\cdots	$P_{1,64}^1$	$P_{2,1}^1$	$P_{2,2}^1$	\cdots	$P_{2,64}^1$	\vdots	\vdots	\ddots	\vdots	$P_{241,1}^1$	$P_{241,2}^1$	\cdots	$P_{241,64}^1$	$P_{1,1}^1$	$P_{1,2}^1$	\cdots	$P_{1,64}^1$	$P_{2,1}^1$	$P_{2,2}^1$	\cdots	$P_{2,64}^1$	\vdots	\vdots	\ddots	\vdots	$P_{241,1}^1$	$P_{241,2}^1$	\cdots	$P_{241,64}^1$	$P_{1,1}^2$	$P_{1,2}^2$	\cdots	$P_{1,64}^2$	$P_{2,1}^2$	$P_{2,2}^2$	\cdots	$P_{2,64}^2$	\vdots	\vdots	\ddots	\vdots	$P_{241,1}^2$	$P_{241,2}^2$	\cdots	$P_{241,64}^2$	$P_{1,1}^3$	$P_{1,2}^3$	\cdots	$P_{1,64}^3$	$P_{2,1}^3$	$P_{2,2}^3$	\cdots	$P_{2,64}^3$	\vdots	\vdots	\ddots	\vdots	$P_{241,1}^3$	$P_{241,2}^3$	\cdots	$P_{241,64}^3$	$P_{1,1}^{44}$	$P_{1,2}^{44}$	\cdots	$P_{1,64}^{44}$	$P_{2,1}^{44}$	$P_{2,2}^{44}$	\cdots	$P_{2,64}^{44}$	\vdots	\vdots	\ddots	\vdots	$P_{241,1}^{44}$	$P_{241,2}^{44}$	\cdots	$P_{241,64}^{44}$	<table border="1" style="border-collapse: collapse; width: 100%; text-align: center;"> <tr><td>$P_{1,1}$</td><td>$P_{1,2}$</td><td>\cdots</td><td>$P_{1,64}$</td></tr> <tr><td>$P_{2,1}$</td><td>$P_{2,2}$</td><td>\cdots</td><td>$P_{2,64}$</td></tr> <tr><td>\vdots</td><td>\vdots</td><td>\ddots</td><td>\vdots</td></tr> <tr><td>$P_{10604,1}$</td><td>$P_{10604,2}$</td><td>\cdots</td><td>$P_{10604,64}$</td></tr> </table>	$P_{1,1}$	$P_{1,2}$	\cdots	$P_{1,64}$	$P_{2,1}$	$P_{2,2}$	\cdots	$P_{2,64}$	\vdots	\vdots	\ddots	\vdots	$P_{10604,1}$	$P_{10604,2}$	\cdots	$P_{10604,64}$
$P_{1,1}^1$	$P_{1,2}^1$	\cdots	$P_{1,64}^1$																																																																																														
$P_{2,1}^1$	$P_{2,2}^1$	\cdots	$P_{2,64}^1$																																																																																														
\vdots	\vdots	\ddots	\vdots																																																																																														
$P_{241,1}^1$	$P_{241,2}^1$	\cdots	$P_{241,64}^1$																																																																																														
$P_{1,1}^1$	$P_{1,2}^1$	\cdots	$P_{1,64}^1$																																																																																														
$P_{2,1}^1$	$P_{2,2}^1$	\cdots	$P_{2,64}^1$																																																																																														
\vdots	\vdots	\ddots	\vdots																																																																																														
$P_{241,1}^1$	$P_{241,2}^1$	\cdots	$P_{241,64}^1$																																																																																														
$P_{1,1}^2$	$P_{1,2}^2$	\cdots	$P_{1,64}^2$																																																																																														
$P_{2,1}^2$	$P_{2,2}^2$	\cdots	$P_{2,64}^2$																																																																																														
\vdots	\vdots	\ddots	\vdots																																																																																														
$P_{241,1}^2$	$P_{241,2}^2$	\cdots	$P_{241,64}^2$																																																																																														
$P_{1,1}^3$	$P_{1,2}^3$	\cdots	$P_{1,64}^3$																																																																																														
$P_{2,1}^3$	$P_{2,2}^3$	\cdots	$P_{2,64}^3$																																																																																														
\vdots	\vdots	\ddots	\vdots																																																																																														
$P_{241,1}^3$	$P_{241,2}^3$	\cdots	$P_{241,64}^3$																																																																																														
$P_{1,1}^{44}$	$P_{1,2}^{44}$	\cdots	$P_{1,64}^{44}$																																																																																														
$P_{2,1}^{44}$	$P_{2,2}^{44}$	\cdots	$P_{2,64}^{44}$																																																																																														
\vdots	\vdots	\ddots	\vdots																																																																																														
$P_{241,1}^{44}$	$P_{241,2}^{44}$	\cdots	$P_{241,64}^{44}$																																																																																														
$P_{1,1}$	$P_{1,2}$	\cdots	$P_{1,64}$																																																																																														
$P_{2,1}$	$P_{2,2}$	\cdots	$P_{2,64}$																																																																																														
\vdots	\vdots	\ddots	\vdots																																																																																														
$P_{10604,1}$	$P_{10604,2}$	\cdots	$P_{10604,64}$																																																																																														

columns (see Fig. 7) and processed in batch mode per i -entry. Since the main interest here is to predict the cumulative oil production, the output node for the i -th entry is defined by

$$V_i = \sum_{k=0}^i y(t_k), \quad (14)$$

where V_i represents the cumulative oil volume obtained until the time t_i . Figure 8 shows an adaption of the architecture previously shown in Fig. 4 to our particular case of study:

The activation functions used were the hyperbolic tangent, defined by

$$f(\mu) = \frac{e^\mu - e^{-\mu}}{e^\mu + e^{-\mu}}, \quad (15)$$

and the rectified linear unit (ReLU), defined by

$$f(\mu) = \max(0, \mu). \quad (16)$$

Some variations of ReLU were considered for testing the hyperparameters, such as the leaky ReLU (Maas et al. 2013), given by as

Fig. 8 Multi-layer perceptron network used to process the i -th row of the input matrix in batch mode: input nodes (a 64-dimensional array); hidden nodes (5 layers \times 51 neurons); output node (scalar value)

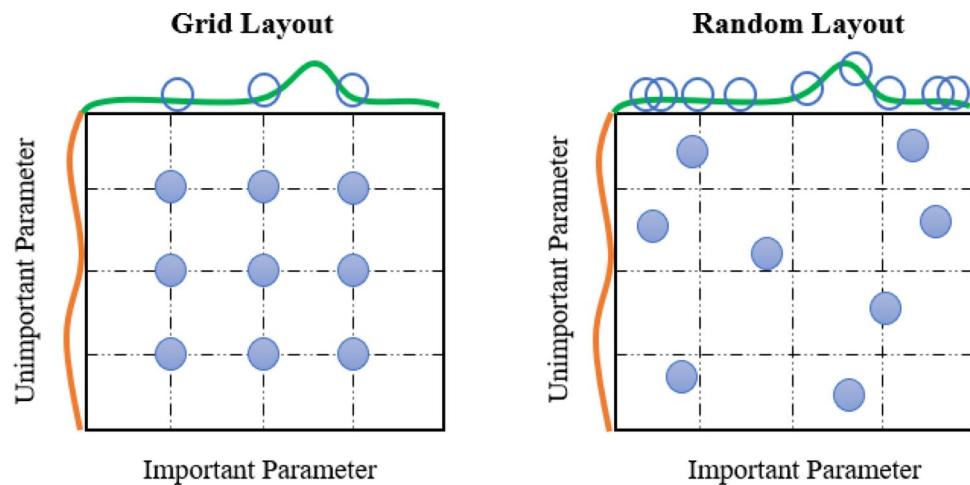
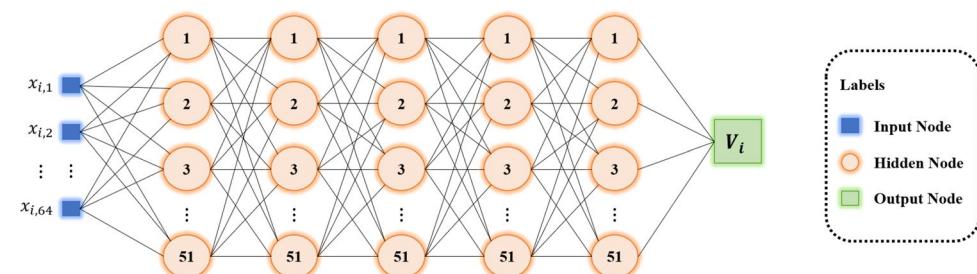


Fig. 9 Comparison of search strategies for two hyperparameters. Both strategies look for a “best-bet” of a pair of important/unimportant hyperparameters in 9 tests. The random layout (on the right side) is capable to generate a better performance of the network (green curve) compared to the grid layout (on the left side). In the grid layout search, the choice of a determined pair is specified by the user.

$$f(\mu) = \max(\alpha\mu, \mu) \quad \alpha \in \mathbb{R}, \quad (17)$$

and the exponential linear unit (ELU) (Clevert et al. 2015), given by

$$f(\mu) = \begin{cases} \mu, & \mu > 0 \\ \alpha(e^\mu - 1), & \mu \leq 0 \end{cases} \quad (18)$$

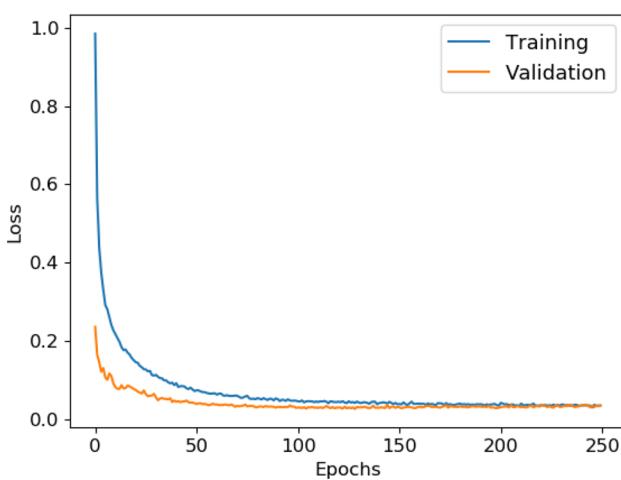
To find suitable hyperparameters, we applied a random search strategy, since this approach is a better choice in relation to grid search which may increase the network’s ultimate performance (Bergstra and Bengio 2012). Figure 9 illustrates a comparison between both for 9 tests.

The configuration defined for the MLP is summarized in Table 1. The weights were initially normalized (Glorot and Bengio 2010) and the backpropagation algorithm was carried with an Adam-like optimizer (Kingma and Ba 2014).

Yet the random layout search, the pairs are selected at random initially and posterior tests are carried out to reduce the range of hyperparameters toward the best response. However, it is proven that the random layout results in better responses due to its wider coverage of the hyperparameter space. Design is based on Bergstra and Bengio (2012)

Table 1 Summary of the MLP configuration used in this paper

Hyperparameter	Option
Number of hidden layers	5
Number of neurons per layer	51
Activation function for hidden layers	ELU
Activation function for output layer	Linear
Optimizer	Adam
Dropout probability	0.08612

**Fig. 10** Loss function curves for training data (blue) and validation data (orange) of the MLP

Results

Cumulative oil production prediction

MLP was trained within 250 epochs under the hyper-parameter values listed in Table 1 by using 44 wells for training and 5 for validation. Figure 10 illustrates the loss function curves for training (blue) and validation (orange). As seen, no overfitting was detected.

The testing set consists of 5 wells located in different HFUs with different DRT values. Their (x, y) positions within the UNISIM-ID reservoir are (44, 20), (24, 40), (52, 11), (59, 14), and (35, 31), with respective DRT values 16, 18, 19, 20, and 21.

Figure 11 shows comparisons between the cumulative oil production yielded by the simulator (curves in black) and the predictions made by MLP (curves in red) for 241 months per well. We note that the oil production estimated by the neural network accompanied the trends of the simulated one in all cases although a few oscillations have appeared. Such oscillations break the always-growing trend originally expected so that the function is non-increasing anymore. To fix them, we post-processed the curves generated by

MLP by pulling up those points where there are negative slopes to ensure at least a monotonic growth. This change will occur for time instants t_{k+1} whenever $\tilde{y}_{k+1} < \tilde{y}_k$, where $\tilde{y}_{k+1} = \tilde{y}(t_{k+1})$ and $\tilde{y}_k = \tilde{y}(t_k)$. If so, the “pull-up” will result in $\tilde{y}_{k+1} = \tilde{y}_k$.

To further verify overestimation and underestimation of the original MLP profiles, integral relative errors were plotted in the form of plateaus scaled at a secondary axis over 60-month (5-year) periods in Fig. 11. With identifying each period by T_5^q , the following formula computes the error V_q :

$$V_q = \int_{T_5^q} \frac{\mathbf{y}(t) - \tilde{\mathbf{y}}(t)}{\mathbf{y}(t)} \approx \sum_{k=60(q-1)}^{60q} \frac{y(t_k) - \tilde{y}(t_k)}{y(t_k)}, \quad q = 1, 2, 3, 4. \quad (19)$$

This way, equal-sized intervals of production can be taken into account to check the fidelity of the network in representing the simulator outputs.

Table 2 lists RMSE, MASE, and error ratios to measure the MLP performance with and without post-processing for the 5 testing wells. Respectively, the subscripts o and p reflect the errors for the original MLP results and post-processed MLP results. The error ratios are defined as

$$\gamma = 1 - \frac{\text{RMSE}_p}{\text{RMSE}_o} \quad (20)$$

$$\rho = 1 - \frac{\text{MASE}_p}{\text{MASE}_o} \quad (21)$$

and approximated to 3 digits. Positive values of γ and ρ mean how much the post-processing improved the original prediction, whereas negative values mean the opposite. As observed, the production at the wells (24, 40) and (35, 31) had considerable improvement. On the other hand, the production at the wells (52, 11) and (59, 14) had a slight worsening. The well (44, 20) was the only one that bypassed post-processing. In turn, $\gamma = \rho = 0.00$, since RMSE_p and MASE_p were kept with same values as original MLP's.

Best-producing well locations

The objective here was to identify the well locations having the best oil production volumes. Given that the MLP performance was previously verified for the testing set only, we now handle the entire data set to look for the best-producing well placement. Table 3 ranks the best productions found. The productions associated to the testing set wells are boldfaced to highlight their recovery potential in relation to all the elements of the data set. Among the ten first wells, only (35,31) is a member of the testing set. Moreover, it coincidentally occupies the first position in the ranking. Other testing wells appear in lower positions. Since

Fig. 11 Comparison of simulated and predicted curves. On the left column, the CMG Imex and MLP curves are plotted together. On the right column, the post-processed points are marked in blue circles over the MLP curves. As seen, the post-processing is minimally required. Dashed segments indicate overestimation (green color) or underestimation (yellow color) relative errors V_q over 5-year periods scaled at the secondary axis at right

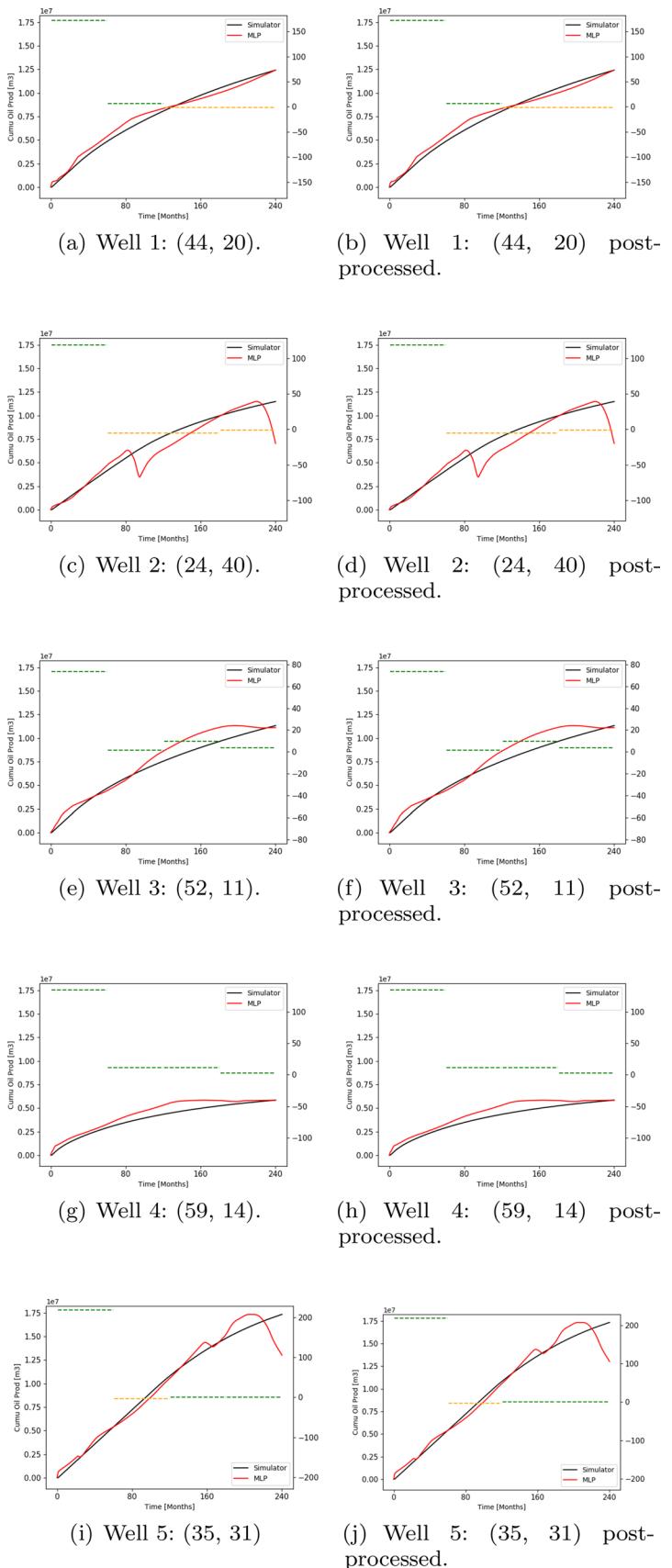


Table 2 Quantitative analysis of original and post-processed predictions of the neural network

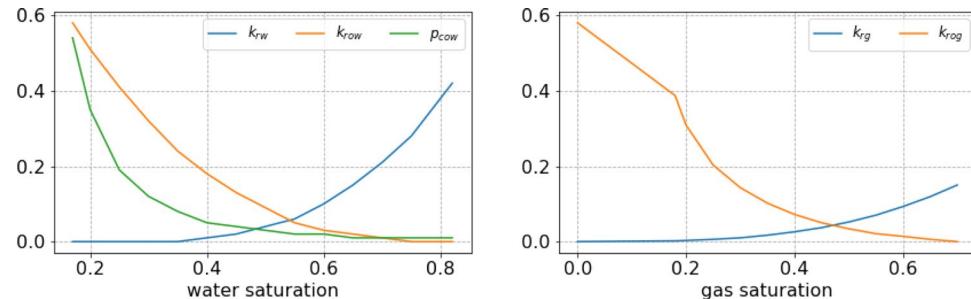
Well	Original MLP		Post-Processed MLP		Error ratios	
	RMSE _o	MASE _o	RMSE _p	MASE _p	γ	ρ
(44, 20)	0.0375	8.0097	0.0375	8.0097	0.00	0.00
(24, 40)	0.0914	15.2439	0.0524	9.8874	0.43	0.35
(52, 11)	0.0804	15.7107	0.0811	16.0941	-0.01	-0.02
(59, 14)	0.1074	22.4529	0.1090	23.1409	-0.01	-0.03
(35, 31)	0.0548	8.5916	0.0345	6.6312	0.37	0.23

Table 3 Ranking of the best producer wells at the flow unit columns found over the UNISIM-I-D model. The maximum oil production is given in cubic meters and the MLP's apex month indicates the month where the network's prediction reached that maximum. Italic faced data correspond to the testing set wells.

Ranking	Well	Max. oil production [m ³]	MLP's apex [month]
1	(35, 31)	17,352,820	206
2	(36, 31)	16,909,378	228
3	(25, 25)	15,530,890	240
4	(49, 20)	15,457,309	219
5	(50, 21)	14,810,013	240
6	(27, 22)	14,799,173	219
7	(35, 29)	14,304,587	240
8	(39, 29)	14,149,472	240
9	(51, 20)	13,873,899	163
10	(37, 34)	13,868,568	236
:			
16	(44, 20)	12,428,663	240
:			
25	(24, 40)	11,489,793	220
26	(52, 11)	11,329,837	197
:			
45	(59, 14)	5,841,570.5	164

the maximum oil production resulting from the network's prediction does not necessarily matches the last month as the simulator does, the further right column in the table indicates the specific month where the network reached the production peak.

Fig. 12 Relative permeability curves used in the reservoir modelling



Conclusion

This paper used the multilayer perceptron (MLP) neural network to predict the cumulative oil production in wells located within hydraulic flow units by learning how oil saturation, pressure, permeability, and porosity behave in the reservoir. The oil recovery predicted by the MLP was compared to the Imex simulator. We noted that the network estimates follow the trend of the curves obtained by the simulator with reasonable average errors. Post-processing was done on the predictions obtained by the MLP to rebalance the production curve at decay moments, which are characterized as misfits. Thus, the averages of the RMSE and MASE error measures for the post-processed MLP overcame those reached from the original outputs.

Regarding the optimal location of the wells, the proposed methodology identified the same locations as those given by the simulator for which the production was maximized. We have shown that the highest oil productions can be ranked properly to specify the most profitable locations.

We ascertain that MLPs are efficient mechanisms to provide reasonable estimates of the oil volume producible by a petroleum reservoir. Furthermore, they advocate for reducing the exhaustive effort with numerical simulations.

Appendix: Reservoir modelling setup

Relative permeability curves

See Fig. 12.,

Fluid characterization and PVT data

Tables 4 and 5.

Table 4 Main properties of fluid characterization used in the reservoir modelling. From left to right, the properties are, in this order: water formation volume factor, oil density, water density, and water compressibility

BWI (m^3/m^3)	ρ_o (kg/m^3)	ρ_w (kg/m^3)	c_w ($1/\text{kg}/\text{cm}^2$)
1.0210	866	1010	47.64E-6

Table 5 PVT data used in the reservoir modelling. From left to right, the properties are, respectively: pressure, solution gas ratio, oil formation volume factor, gas formation volume factor, oil viscosity, gas viscosity, and oil compressibility

p (kg/cm^2)	R_s (m^3/m^3)	B_o (m^3/m^3)	B_g (m^3/m^3)	μ_o (mPa.s)	μ_g (mPa.s)	c_o ($1/\text{kg}/\text{cm}^2$)
35.49	31.8	1.198	0.0346	2.05	0.0109	1.62E-4
41.82	34.66	1.2	0.0291	1.99	0.0113	1.62E-4
49.2	38.02	1.21	0.0245	1.91	0.0117	1.62E-4
59.75	42.83	1.23	0.0199	1.81	0.0123	1.62E-4
68.54	46.85	1.24	0.0172	1.73	0.0128	1.62E-4
80.85	52.51	1.25	0.0144	1.62	0.0134	1.62E-4
93.86	58.51	1.27	0.0123	1.52	0.0142	1.62E-4
105.81	64.06	1.28	0.0108	1.43	0.0148	1.62E-4
121.98	71.6	1.3	0.0093	1.32	0.0157	1.62E-4
133.94	77.2	1.32	0.0084	1.25	0.0164	1.62E-4
148.0	83.83	1.33	0.0076	1.17	0.0172	1.62E-4
166.29	92.49	1.35	0.0067	1.09	0.0182	1.62E-4
193.36	105.42	1.39	0.0058	1.0	0.0197	1.62E-4
213.26	115.01	1.41	0.0053	0.96	0.0208	1.62E-4
219.38	117.64	1.42	0.0051	0.94	0.0211	1.62E-4
229.5	122.19	1.43	0.0049	0.91	0.0217	1.62E-4
248.0	130.84	1.45	0.0045	0.85	0.0227	1.62E-4
283.02	147.22	1.5	0.004	0.75	0.0246	1.62E-4
316.91	163.08	1.54	0.0035	0.65	0.0265	1.62E-4
352.63	179.79	1.58	0.0032	0.54	0.0285	1.62E-4
360.0	183.24	1.59	0.0031	0.52	0.0289	1.62E-4

Acknowledgements E.F.M.N. acknowledges the support of the CNPq-Brazil's scholarship program. G.P.O and M.D.S. thank Petrobras and the Brazilian National Agency of Petroleum, Natural Gas, and Biofuels (ANP) for the funding (R&D project no. 2018/00051-8). L.V.B. also gratefully acknowledges the support of NVIDIA Corporation with the donation of the Titan Xp GPU card used for this research.

Open access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

Declarations

Conflict of interest The authors state that there is no conflict of interest.

Ethical statement The authors state that there is no ethical conflict with sources of funding.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Aizenberg I, Sheremetov L, Villa-Vargas L, Martinez-Muñoz J (2016) Multilayer neural network with multi-valued neurons in time series forecasting of oil production. Neurocomputing 175:980. <https://doi.org/10.1016/j.neucom.2015.06.092>
- Amaefule JO, Altunbay M, Tiab D, Kersey DG, Keelan DK, et al. (1993) in SPE annual technical conference and exhibition (Society of Petroleum Engineers, 1993). <https://doi.org/10.2118/26436-MS>
- Avansi GD, Schiozer DJ (2015) Unisim-i: synthetic model for reservoir development and management applications. Int J Model Simul Petroleum Indus 9(1):21
- Bergstra J, Bengio Y (2012) Random search for hyper-parameter optimization. J Mach Learn Res 13(1):281
- Bernetti SM, Shahbazian M (2011) An imperialist competitive algorithm artificial neural network method to predict oil flow rate of the wells. Int J Computer Appl 26(10):47. <https://doi.org/10.5120/3137-4326>
- Chakra NC, Song KY, Gupta MM, Saraf DN (2013) An innovative neural forecast of cumulative oil production from a petroleum reservoir employing higher-order neural networks (honns). J Petroleum Sci Eng 106:18. <https://doi.org/10.1016/j.petrol.2013.03.004>
- Clevert DA, Unterthiner T, Hochreiter S (2015) Fast and accurate deep network learning by exponential linear units (elus), arXiv preprint [arXiv:1511.07289](https://arxiv.org/abs/1511.07289)
- Glorot X, Bengio Y (2010) Proceedings of the thirteenth international conference on artificial intelligence and statistics 249–256
- Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT press, USA
- Guo G, Diaz M, Paz F, Smalley J, Waninger E, et al. (2005) in SPE Annual Technical Conference and Exhibition (Society of Petroleum Engineers, 2005). <https://doi.org/10.2118/97033-MS>
- Hu H, Zhai X, Feng J, Guan X (2018) in 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS) (IEEE, 2018), pp. 267–270. <https://doi.org/10.1109/ICSESS.2018.8663751>
- Hyndman RJ, Koehler AB (2006) Another look at measures of forecast accuracy. Int J Forecast 22(4):679. <https://doi.org/10.1016/j.ijforecast.2006.03.001>
- Kingma DP, Ba J (2014) Adam: A method for stochastic optimization, [arXiv:1412.6980](https://arxiv.org/abs/1412.6980)
- Liu W, Liu WD, Gu J (2020) Forecasting oil production using ensemble empirical model decomposition based long short-term memory neural network. J Petroleum Sci Eng 189:107013. <https://doi.org/10.1016/j.petrol.2020.107013>
- Maas AL, Hannun AY, Ng AY, (2013) in Proc. icml, vol. 30 (2013), vol. 30
- Mamudu A, Khan F, Zendehboudi S, Adedigba S (2020) Dynamic risk assessment of reservoir production using data-driven probabilistic approach. J Petroleum Sci Eng 184:106486. <https://doi.org/10.1016/j.petrol.2019.106486>
- Min B, Park C, Kang J, Park H, Jang I (2011) Optimal well placement based on artificial neural network incorporating the productivity potential. Energy Sour, Part A: Recovery, Utilization, Environ Eff 33(18):1726. <https://doi.org/10.1080/15567030903468569>
- Oliveira G, Roque W, Araújo E, Diniz A, Simões T, Santos M (2016) Competitive placement of oil perforation zones in hydraulic flow units from centrality measures. J Petroleum Sci Eng 147:282. <https://doi.org/10.1016/j.petrol.2016.06.008>
- Oliveira G, Santos M, Roemers-Oliveira E (2020) Well placement sub-clustering within partially oil-saturated flow units. J Petroleum Sci Eng 196:107730
- Roque W, Oliveira G, Santos M, Simões T (2017) Production zone placements based on maximum closeness centrality as strategy for oil recovery. J Petroleum Sci Eng 156:430
- Sagheer A, Kotb M (2019) Time series forecasting of petroleum production using deep lstm recurrent networks. Neurocomputing 323:203. <https://doi.org/10.1016/j.neucom.2018.09.082>
- Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (1929) Dropout: a simple way to prevent neural networks from overfitting. J Machine Learn Res 15(1):356

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.