19th Brazilian Congress of Thermal Sciences and Engineering
November 6th–10th, 2022, Bento Gonçalves - RS - Brazil

ENC-2022-0233
ESTIMATION OF WELL PRODUCTION FLOW RATE USING RECURSIVE NEURAL NETWORKS

Rafael Bastos Scisinio Dias
Sergio Santiago Ribeiro
Márcio da Silveira Carvalho
Mechanical Engineering Department, Pontifícia Universidade Católica do Rio de Janeiro, RJ 22451-900, Brazil
rbsdias22@gmail.com;sergio@lmmp.mec.puc-rio.br;msc@puc-rio.br

Abstract.*Reservoir characterization is a very important task in the oil and gas industry. The common approach is to stop production to perform well tests. During the testing period, the production well is either controlled at a constant flow rate (Draw-down) or it is closed (Buildup). The test can take a few hours up to a few days to generate all data needed to estimate the reservoir parameters based on the transient response of the well-reservoir system. In most of the recently built wells, permanent downhole gauges (PDGs) are being installed, enabling pressure and temperature data collection during production. Most of these sensors do not include flowrate measurements. The ability to estimate flow rate of producing wells based on the transient response of pressure and temperature during production could speed up the estimation of evolution of reservoir parameters and the overall management of the oil field. This estimation can be done by coupling the solution of an inverse problem and a reservoir simulator. This alternative is computationally very expensive and not practical in most applications. An alternative, is to use deep neural networks to estimate production flow rate and perform reservoir properties estimation using the PDG data. In this work, we use a Recursive Neural Network (RNN) to study the behavior of a reservoir with known permeability. The pressure and temperature profiles are generated by a reservoir flow simulator, with a prescribed flow rate time evolution input. Part of this data set is passed as training data to the RNN. Therefore, the trained RNN flow rate predictions for an unseen data set are then compared with the simulated ground truth. Finally, a sensitivity analysis is performed changing the sliding window size, RNN architecture and training parameters. The results obtained showed satisfactory behavior, as well as a reduced error.*

Keywords: *Artificial intelligence, neural network, PDG*

1. INTRODUCTION

The characterization of oil reservoir flow properties, as well as the determination of well production are important, as they help exploration strategies optimization for an oil field (Tian, 2018; Garuzzi and Romero, 2014). Since detailed well data is difficult to obtain and expensive with current methods, the application of artificial intelligence techniques to estimate the production flow rate of wells has become an alternative. However, it is still unclear whether different techniques based on artificial intelligence, such as machine learning, deep learning and neural networks, are capable of estimating well production flow.

Therefore, the present work aims to present a formulation and elaborate a computational code, using artificial intelligence techniques, which is capable of estimating the production flow of wells, with a low level of error. For this, some neural network models were tested, and the generated predictions were compared with the real data.

2. THEORETICAL BACKGROUND

2.1 Well Testing

Reservoir characterization is a very important task for the oil and gas sector, and one of the most used approaches for this is well testing. During the testing period, the pressure evolution is measured during a period that the well is operated either at a constant flow rate (Drawdown Test) or closed (Build-up Test). and the duration of the test can last from a few hours to a few days (Tian, 2018)

In the Drawdown test, the well is opened for production, at a constant flow rate other than zero, causing the pressure to drop in the reservoir over time, as shown in Figure 1. The objectives of this test are: obtaining the average permeability of the rock, determining the volume of pores, detect the heterogeneity of the reservoir among others. On the other hand, in the Build-up test, the well is closed (shut-in or zero flowrate), causing the reservoir pressure to increase over time, as shown in Figure 2. The objectives of the test are: to determine the static pressure of the reservoir, determine the effective permeability of the reservoir, the limits of the reservoir, among others.

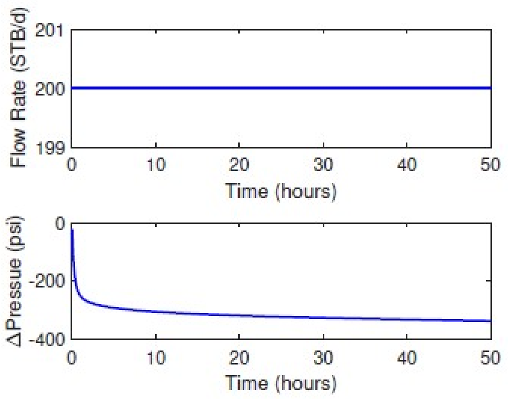

Figure 1. Drawdown test example (Tian, 2018).

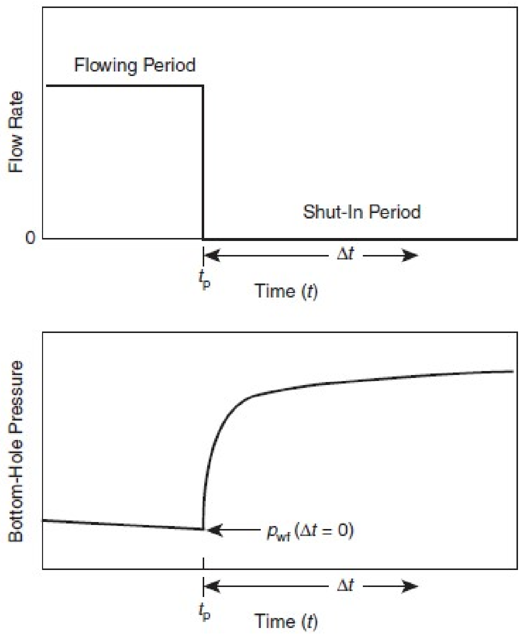

Figure 2. Buildup test example (Meehan, 2012)

Hence, the pressure data collected in the tests are used to characterize several parameters of the reservoir such as: reservoir boundaries, permeability and well storage.

2.2 Permanent Downhole Gauge (PDG)

To measure pressure, temperature and, in some cases, the production flow, a device called PDG (Permanent Downhole Gauge) is installed in the well, represented in Figure 3, which provides continuous data on the aforementioned quantities and was initially used only for monitoring. of production. However, over time, it was realized that the PDG could be used to characterize the reservoir around the well.

However, unlike in the well testing operations, PDGs obtain pressure and temperature under variable flow conditions during the long period of production (Tian, 2018). In addition, in PDGs the flowrate values are not measured, making the techniques applied in well tests not suitable for the analysis of PDG data. Faced with this problem, machine learning techniques, such as neural networks, have been used to analyze this data (Tian and Horne, 2019).

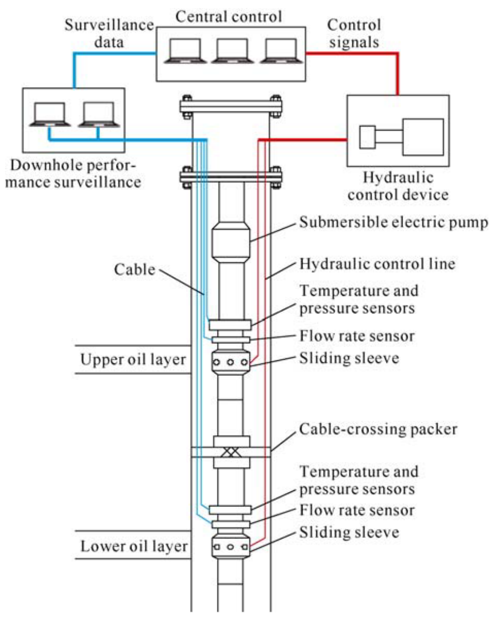

Figure 3. Permanent Downhole Monitoring system (He *et al.*, 2020).

2.3 Artificial Intelligence

The term Artificial Intelligence (AI) was introduced in the 1950s and refers to machines that exhibit human-like behavior. Some of the objectives in AI research are: learning, reasoning, communication (natural language processing), manipulation and movement of objects. And, despite AI being an area of study of computer science, it is currently being applied in several other areas such as robotics, biology, engineering, among others (Ongsulee, 2017; GOMES, 2010).

2.3.1 Machine learning

Machine learning (ML) is a sub-area of artificial intelligence, in which computers are led to build complex mathematical correlations that resemble learning, without the introduction of a direct instruction given by the programmer, such as Artificial Neural Networks (ANN). Overall, the algorithms used in ML are able to use their own errors to evolve these correlations, detect patterns, and continually improve model predictions. They are commonly used in situations where the development of programming code would be very complex, or even impractical (Ongsulee, 2017; Lemley *et al.*, 2017).

Machine learning methods can be divided into supervised, unsupervised and semi-supervised as seen in Nasteski (2017) and Lemley *et al.* (2017). In this work, the method used will be the supervised learning, in which, during training, the known input and output data are provided to the model, so that the algorithm can find the correlations and thus infer output values for a never-before-seen input dataset.

2.3.2 Neural networks

Neural networks, were designed with the aim of modeling the way the brain acts in performing a task, with the behavior of neurons being the main inspiration for the network model (FLECK *et al.*, 2016; Bi *et al.*, 2019). Structurally, networks are composed of layers of artificial neurons, divided into 3 groups: input layer, hidden layer and output layer (Lemley *et al.*, 2017; Hinton *et al.*, 2015; Pacheco and Pereira, 2018).

The conversion of inputs x to outputs y is done in the hidden layers, in which the sum of the Hadamard product, of inputs and weights w is added to a bias (μ) and applied to an activation function f, as seen in Eq. (1).

$$y = f(\sum_{i=1}^{m} w_i x_i - \mu) \tag{1}$$

The activation function processes the generated signal, from the linear combination of weights and inputs, generating the output signal of the artificial neuron (FLECK *et al.*, 2016; Rauber, 2005; Silva, 2010). The type of the activation function may vary with the application, being the sigmoid function the one used in this paper. The sigmoid function is the type of activation function most used in ANNs, since it is a continuous function, with an adequate balance between linear and non-linear behavior. An example of a sigmoid function is the logistic function as in Eq. (2), which has an interval between 0 and 1.

$$f(z) = \frac{1}{1+e^{-z}} \tag{2}$$

In addition to choosing the activation function that best suits the application, evaluating the performance of the neural network is an important metric, as it is a parameter that indicates whether the ANN performs within the desired tolerances, or whether adjustments are necessary. The most used form of evaluation is the Mean Square Error (MSE), represented in Eq. (3) (Silva, 2010).

$$MSE = \frac{1}{N}\sum_{j=1}^{M}\sum_{i=1}^{N}(y_{p_{ji}} - y_{r_{ji}})^2 \tag{3}$$

Where N is the number of training samples, M is the number of outputs from the network, $y_{p_{ji}}$ is the output predicted by the network, $y_{r_{ji}}$ is the actual value of the output and MSE is the root mean square error between the predicted and actual output value. Ideally, the value of the MSE is minimized during training, until it is within the tolerable limits of the project (Silva, 2010).

2.3.3 Deep learning

Deep Learning (DL) is an area of machine learning that refers to artificial neural networks with more than 1 hidden layer. The advantage of deep learning compared to machine learning is the ability to work with raw data, that is, pre-processing the data is not necessary (Ongsulee, 2017; Rusk, 2016; Hinton *et al.*, 2015; Pacheco and Pereira, 2018).

Like machine learning algorithms, deep learning algorithms are used for image recognition, facial recognition, natural language processing, among others. These applications have been used by several companies such as Google, for the development of autonomous cars and translation services (Ongsulee, 2017; Hinton *et al.*, 2015; Pacheco and Pereira, 2018).

2.3.4 Recurrent neural network (RNN)

Recurrent Neural Networks (RNN) are a type of neural network that has data feedback, making this network structure very useful for working with temporal or sequential data (Silva, 2010; Rauber, 2005; Lemley *et al.*, 2017; Hinton *et al.*, 2015). Unlike a Feed Forward Network, in which the information flow is unidirectional, as shown in Figure 4, in RNNs the hidden layers have a feedback, allowing the information to be passed back to the neuron, as seen in Figure 5 (Silva, 2010; Rauber, 2005; Lemley *et al.*, 2017; Hinton *et al.*, 2015; Tian, 2018).

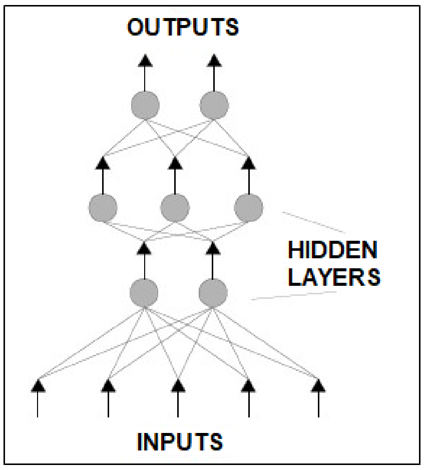

Figure 4. Feedforward network (Rauber, 2005).

As shown in Figure 5, each recurrent unit, for each instant t, calculates the hidden state (h_t) taking into account the input (x_t) and the hidden state of the previous time (h_{t-1}), and calculates the output (y_t). The calculation of the hidden state and the output of the recurring cell are represented in Eq. (4) and Eq. (5). Where W_x, W_h and W_y are the weight matrices of the input, hidden state and output respectively, μ_h and μ_y are the bias vectors and f_1 and f_2 activation functions (Tian, 2018).

$$h_t = f_1(x_t W_x + h_{t-1} W_h + \mu_h) \tag{4}$$

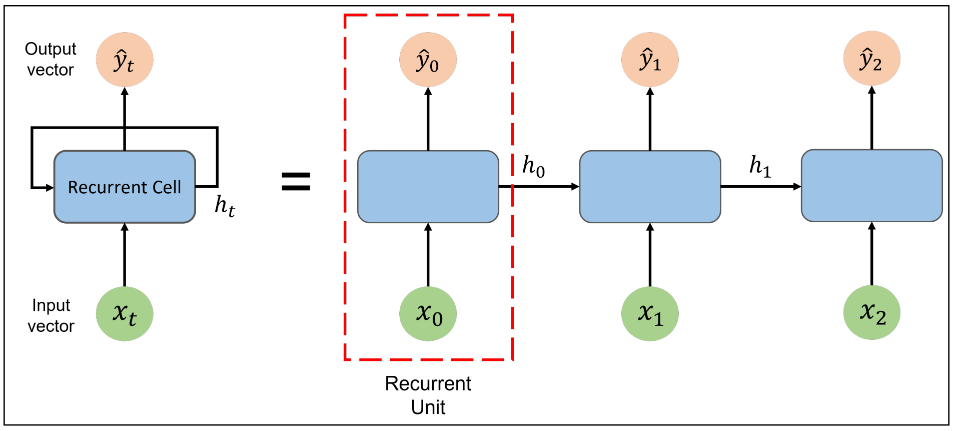

Figure 5. Recurrent neural network (Hinton *et al.*, 2015).

$$y_t = f_2(W_y h_t + \mu_y) \tag{5}$$

The parameters W_x, W_h, W_y, μ_h and μ_y are usually initialized by the user, and the goal is to find the parameter values so that the error is minimized. One of the methods used is the gradient descent in which, through an iterative process, one seeks to find θ, the vector of the model's parameters, which minimizes the error. And θ is calculated by the difference between θ and the product between the learning rate (α) and the gradient of the error function $(\nabla J(\theta))$ with respect to θ, as shown in Eq. (6) (Tian, 2018).

$$\theta = \theta - \alpha \nabla J(\theta) \tag{6}$$

A problem that occurs when training a RNN is the vanishing (or explosion) of the gradient, because to calculate it, a matrix of weights is multiplied by itself several times, due to the application of the chain rule. Thus, if the elements of the matrix are close to zero, the gradient vanishes and if they are very large, the gradient explodes (Tian, 2018). A solution to this problem is the use of more robust and advanced RNN architectures such as Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU).

2.3.5 Long Short-Term Memory

Unlike simple recurrent networks, which are not capable of maintaining long-term dependencies (short memory), long short-term memory units (LSTM), introduced by Hochreiter and Schmidhuber (1997), are a type of RNN capable of storing long-term information. An LSTM unit is able to add or remove information to the cell state, through 3 internal gates (forget gate, input gate and output gate) (Hochreiter and Schmidhuber, 1997; Dutta *et al.*, 2020; Cho *et al.*, 2014; Chung *et al.*, 2015; Rodrigues, 2021).

2.3.6 Gated Recurrent Unit

The Gated Recurrent Unit (GRU) is another type of RNN capable of handling long-term dependencies. Its structure was proposed by Cho *et al.* (2014) and it is similar to an LSTM unit. The GRU also has an internal structure containing gates, but unlike the LSTM which has 3 gates, the GRU has only 2 gates: the update gate, that is a combination of the forget gate and input gate, and the other gate is the reset gate (Cho *et al.*, 2014; Dutta *et al.*, 2020; Chung *et al.*, 2014, 2015).

3. METHODOLOGY

For the simulations carried out in this work, the code from Pedersen (2018) was used as base, modifying the data source, the structure of the neural network, the value of certain parameters of the code, for the construction of the 4 models (LSTM1, LSTM2, GRU1 and GRU2) used in this paper. The codes were written in Python and the Tensorflow library was used to implement the RNN. Other libraries such as: Matplotlib, Numpy, Pandas and Scikit-learn were used for data processing, graphics visualization, among others.

3.1 Data

Two sets of data were used in this work. Sets 1 and 2 were generated from a reservoir dynamics direct simulator that reproduced the behavior of pressure and temperature from a given fictitious flowrate production profile. Figure shows the pressure and temperature resulting curves from one of this simulations, used as input series to the RNN model. Each data set has 105,121 values representing 1 year of temperature, pressure and flowrate.

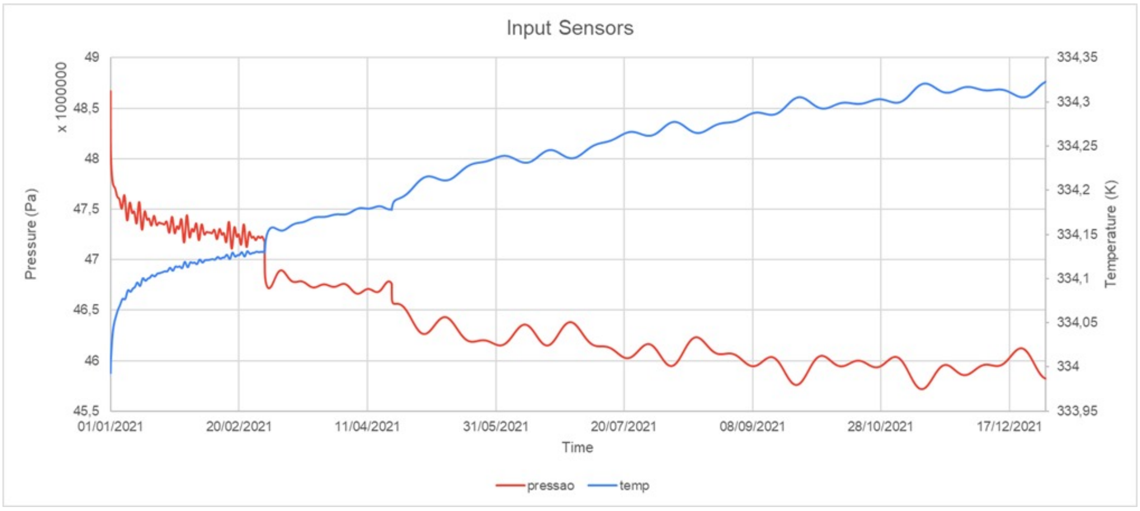

Figure 6. Recurrent neural network (Hinton *et al.*, 2015).

As seen in Table 1, the time values are in seconds, and represent the simulation time for each data. However, as each data is not equally spaced, it was necessary to resample both dataset, so that the new dataset had data at 5 minutes intervals and the new values of flow, temperature and pressure were updated by the average values within 5 minutes, as also seen in Table 1.

Table 1. Original and Resampled data

Original data				**Resampled data**			
Time (s)	**Flow rate (m^3/day)**	**Temp. (K)**	**Pressure (Pa)**	**Time (s)**	**Flow rate (m^3/day)**	**Temp. (K)**	**Pressure (Pa)**
0	0	334	49033000	01/01/2021 00:00	0.009200	333.9939	48670544.52
0.0001025	0.009259	334	49033000	01/01/2021 00:05	0.009259	333.9932	48424272.55
...	...	...	...	...	...	...	...
31535729.7	0.014775	334.3234	45823370.78	31/12/2021 23:55	0.014775	334.3234	45823370.78
31536000	0.014774	334.3235	45823393.59	01/01/2022 00:00	0.014774	334.3235	45823393.59

3.2 Neural network structure

The structure used in all models, as shown in Table 2, was defined based on the values which maximize the use of the available Graphic Processing Unit (GPU) from NVidia, GTX 1070.

Table 2. Neural network structure

Layer	**Type**	**Neurons**
1st	LSTM/GRU	375
2nd	LSTM/GRU	375
3rd	Dense	200
4th	Dense	1

3.3 Parameters

In order to carry out the simulations, in addition to the network structure, some important parameters were defined, to ensure a standardization of the simulations in all models (LSTM 1, LSTM 2, GRU1 and GRU 2). The parameter values are according to Table 3. Where the number of shifted steps represents how many steps in the future the models will predict, the batch size represents how many data sequences (input and output) will be used for training, the sequence length represents how many values each batch will have, the number of epochs represents how many iterations will be done in the batches and the steps per epoch are the number of steps needed to consider an epoch has ended.

Table 3. Code parameters

Parameter	**Value**	**Parameter**	**Value**	**Parameter**	**Value**
Shifted steps	288 (1 day)	Learning rate reducing factor	0.1	Batch size	150
Train split	85(percent)	Min learning rate	1×10^{-6}	Sequence length	1440 (5 days)
Epochs	25	Steps per epoch	300	Learning rate	1×10^{-3}

3.4 Training

All models were trained using 85 (percent) of the datasets, and the models LSTM1 and GRU1 used dataset 1 as a source, and the models LSTM2 and GRU2 used dataset 2.

The network training uses the gradient descent method in order to find the weights that minimize the error (MSE), however the computational effort to calculate the gradient using all the training data (89,000 data) is very high. Therefore, 150 input and output batches were created, starting at random indexes and containing 1440 data, so that the gradient descent method was applied in the batches, reducing the computational effort.

In addition, to avoid that during training of the models, the error stop decreasing or start to increase, the early stopping method was applied. The early stopping method interrupts the training if the error does not decrease in 5 epochs. Another method applied was the reduction of the learning rate (α), which would be multiplied by a factor of 0.1 every time the error does not decrease in 2 epochs, and this reduction is done until α reaches the value of 1×10^{-6}.

To ensure that the models always use the most optimized weights to date, the Checkpoint method was used, so that only the weights that produce the best performance (lowest error) are saved. That way, even if the error increases between 2 epochs, the optimal weights will not change.

3.5 Validation and extension test

For validation of the trained models, the remaining 15 (percent) of each dataset was used as test data. These values were not used or visualized by the models during training. Therefore, each model received the respective input test data, generated the output of each model that were compared with the output test data, resulting in an MSE value of the test data.

To verify the extension of the models corresponds to evaluate their performance predicting datasets with a different behavior, when compared to the one used in training. To verify the extent of models LSTM1/GRU1 dataset 2 was used, and analogously the dataset 1 was used to verify the extent of LSTM2/GRU2. Similarly to the validation data, the extension input data was passed to the respective models, and the generated outputs compared to the respective extension output data, resulting in a MSE value of the extension test.

3.6 Feedback prediction

For the feedback predictions, an iterative procedure was implemented, in which a moving window of 1440 data was used as input for the models forecasting. In the first iteration, the last 1440 training data were used to predict the 1^{st} test value. Then, the window advances one index in the process of generating the forecast, removing the oldest flow data from the input and adding the last predicted value as the most recent input flow. The process repeats until all test data points were predicted.

That analysis is very important, because it mimics the real production forecast process for oil-field planning. This because, the future production values are not known yet, so the predicted values have to be used in the forecast of a larger future prediction window.

4. RESULTS

To evaluate and compare the results of each model, data is divided into 3 parts: validation data, extension test data and feedback forecast data. The output forecasts of each model as ploted against the real production values, and the mean squared errors (MSE) are calculated.

4.1 Validation

The predictions errors (MSE) of the of each model, with respect to their corresponding validation datasets, are shown in Table 4.

Table 4. MSE validation set

Model	MSE
LSTM1	$6.178{\times}10^{-4}$
LSTM2	$6.434{\times}10^{-4}$
GRU1	$7.726{\times}10^{-4}$
GRU2	$7.150{\times}10^{-4}$

From the data in Table 4, it can be observed that all the results are in the same order of magnitude, but the LSTM models presented slightly better performance. Figures 7 and 8 show a sample of the predictions of each model, compared with respective True flow data. It is possible to notice that all models were able to reproduce tendency of the validation data.

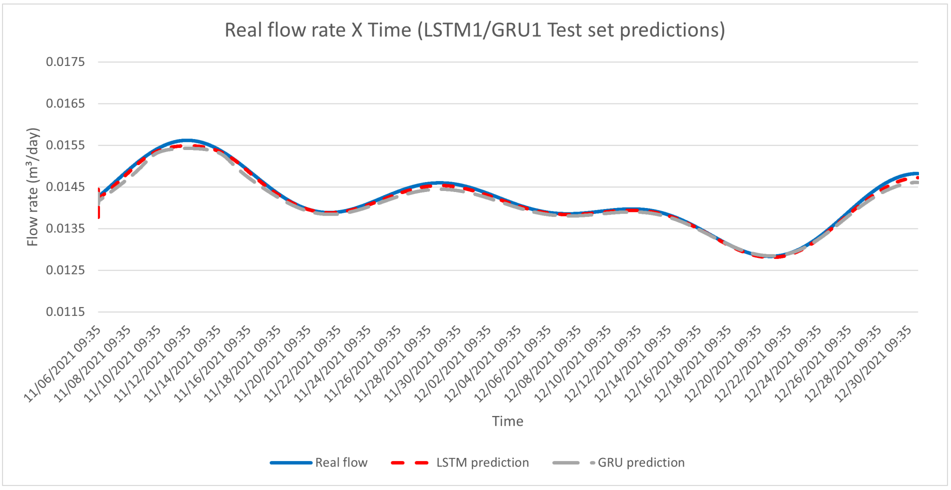

Figure 7. Real flow rate X Time (LSTM1/GRU1 Test set predictions).

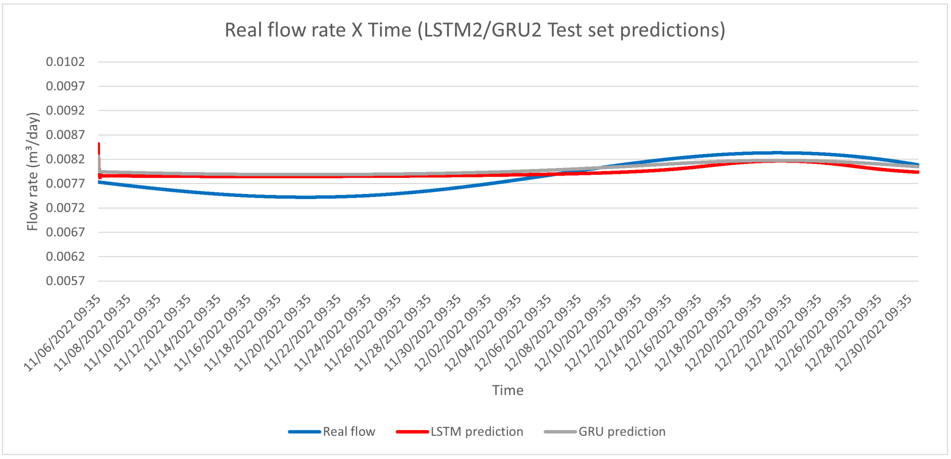

Figure 8. Real flow rate X Time (LSTM2/GRU2 Test set predictions).

4.2 Extension tests

The extension test, or Verification of coverage, of a neural network indicates whether it is able, when receiving unknown data to maintain the performance regarding the validation data. The forecast error (MSE) of each model for the respective coverage data, are shown in Table 5.

Table 5. MSE extension set

Model	MSE
LSTM1	2.119×10^{-2}
LSTM2	2.655×10^{-3}
GRU1	1.146×10^{-2}
GRU2	2.846×10^{-3}

From the data in Table 5, it is possible to identify that the models trained with the 2^{nd} set of data had a better performance than those trained with the 1^{st} set, yet all models presented a satisfactory performance.

The outputs (forecasts) for the coverage data are shown in Figure 9 and Figure 10 in comparison with the respective ground truth. It is possible to notice that the LSTM 1 and GRU 1 models presented some discrepancies in the initial and final parts of the forecasts, as seen in Figure 9. The same did not occur with the outputs of the LSTM 2 and GRU 2 models, as shown in Figure 10.

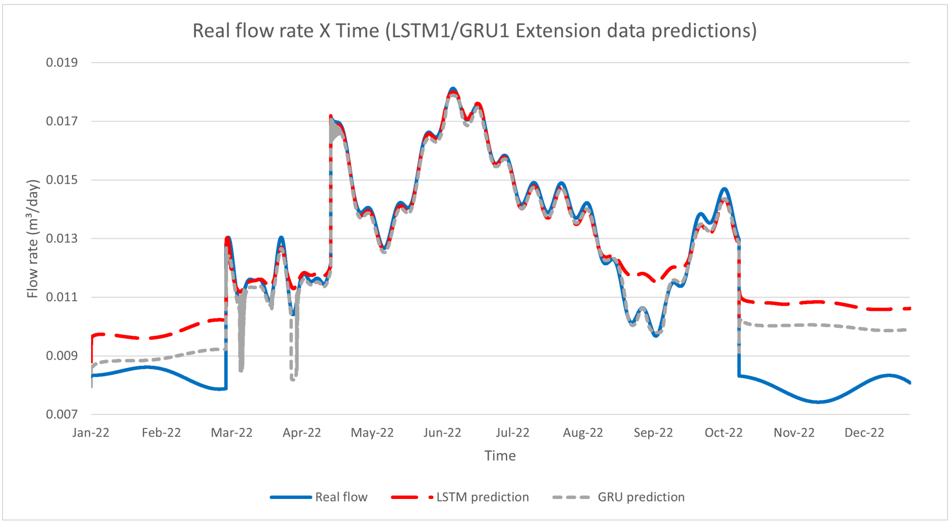

Figure 9. Real flow rate X Time (LSTM1/GRU1 Extension data predictions).

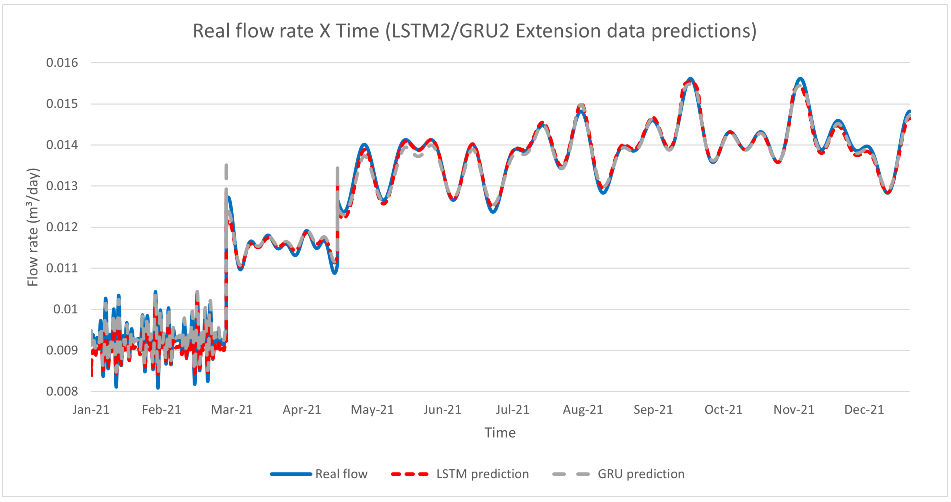

Figure 10. Real flow rate X Time (LSTM2/GRU2 Extension data predictions).

4.3 Feedback data

The results (MSE) of the feedback forecasts of each model, in relation to the respective test data (validation) are shown in Table 6. The visualization of the feedback forecasts are as shown in Figure 11 and Figure 12. It is possible to notice that all models were able to reproduce the behavior of real data, however, the LSTM 1 model presented the worst performance, as also seen in Table 6.

Table 6. MSE forecast set

Model	MSE
LSTM1	$2.228{\times}10^{-2}$
LSTM2	$5.754{\times}10^{-5}$
GRU1	$1.804{\times}10^{-4}$
GRU2	$4.559{\times}10^{-5}$

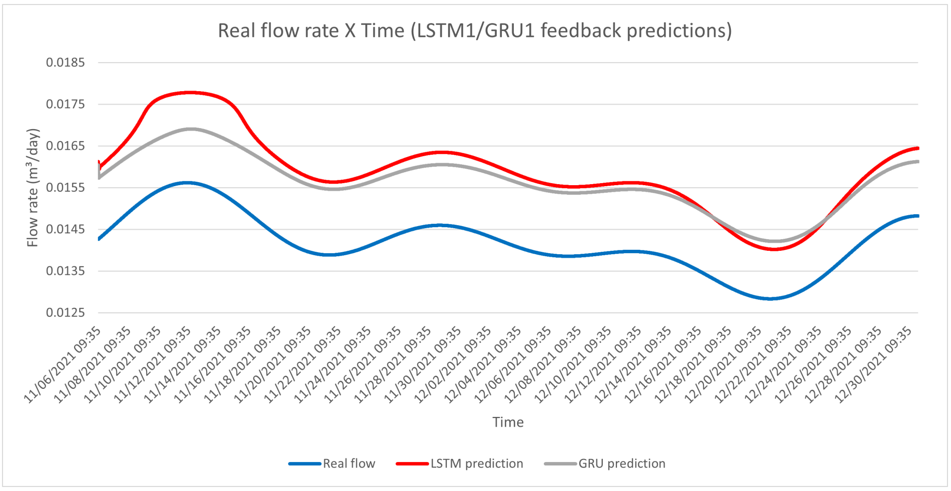

Figure 11. Real flow rate X Time (LSTM1/GRU1 feedback predictions).

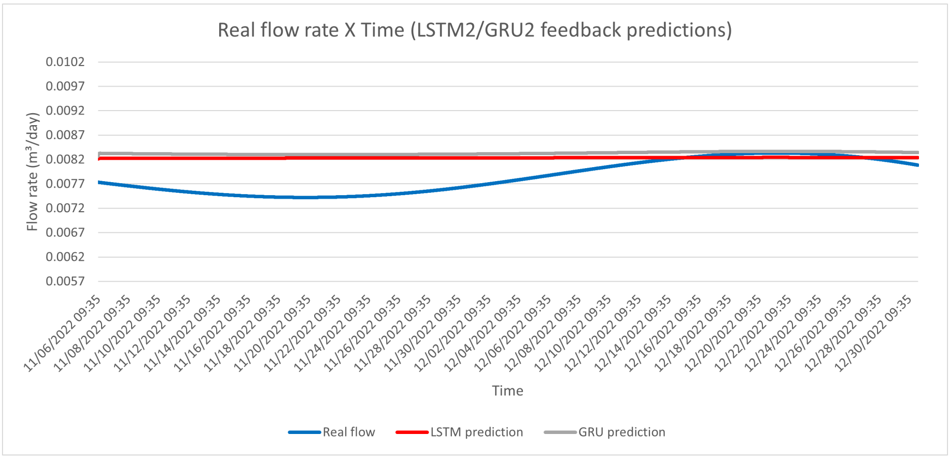

Figure 12. Real flow rate X Time (LSTM2/GRU2 feedback predictions).

5. CONCLUSIONS

The present work sought to develop a model capable of estimating the production flow of wells using artificial intelligence techniques. In view of the results obtained, it is possible to conclude that the 4 network models elaborated (LSTM 1, LSTM 2, GRU 1 and GRU 2) were able to estimate the production flow with a considerably reduced level of error.

In addition, the performance of the networks was very similar, indicating that, in this case, the type of RNN (LSTM or GRU) had little effect on performance. But it is worth noting that, as the training was done using random starting points, when performing other simulations with the same codes, the results may vary, both positively and negatively.

6. REFERENCES

Bi, Q., Goodman, K.E., Kaminsky, J. and Lessler, J., 2019. "What is machine learning? a primer for the epidemiologist". *American journal of epidemiology*, Vol. 188, No. 12, pp. 2222–2239.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y., 2014. "Learning phrase representations using rnn encoder-decoder for statistical machine translation". *arXiv preprint arXiv:1406.1078*.

Chung, J., Gulcehre, C., Cho, K. and Bengio, Y., 2014. "Empirical evaluation of gated recurrent neural networks on sequence modeling". *arXiv preprint arXiv:1412.3555*.

Chung, J., Gulcehre, C., Cho, K. and Bengio, Y., 2015. "Gated feedback recurrent neural networks". In *International conference on machine learning*. PMLR, pp. 2067–2075.

Dutta, A., Kumar, S. and Basu, M., 2020. "A gated recurrent unit approach to bitcoin price prediction". *Journal of Risk and Financial Management*, Vol. 13, No. 2, p. 23.

FLECK, L., Tavares, M.H.F., Eyng, E., Helmann, A. and Andrade, M.d.M., 2016. "Redes neurais artificiais: Princípios básicos". *Revista Eletrônica Científica Inovação e Tecnologia*, Vol. 1, No. 13, pp. 47–57.

Garuzzi, R.P. and Romero, O.J., 2014. "Abordagem analítica e computacional do teste drawdown". *Latin American Journal of Energy Research*, Vol. 1, No. 1, pp. 39–45.

GOMES, D.d.S., 2010. "Inteligência artificial: conceitos e aplicações". *Olhar Científico. v1*, , No. 2, pp. 234–246.

He, L., ZHENG, L., Qinghai, Y., Jiaqing, Y., Qingfeng, Y., Deli, J. and Quanbin, W., 2020. "Development and prospect of separated zone oil production technology". *Petroleum Exploration and Development*, Vol. 47, No. 5, pp. 1103–1116.

Hinton, G., LeCun, Y. and Bengio, Y., 2015. "Deep learning". *Nature*, Vol. 521, No. 7553, pp. 436–444.

Hochreiter, S. and Schmidhuber, J., 1997. "Long short-term memory". *Neural computation*, Vol. 9, No. 8, pp. 1735–1780.

Lemley, J., Bazrafkan, S. and Corcoran, P., 2017. "Deep learning for consumer devices and services: Pushing the limits for machine learning, artificial intelligence, and computer vision." *IEEE Consumer Electronics Magazine*, Vol. 6, No. 2, pp. 48–56.

Meehan, D.N., 2012. *Advanced Reservoir Management and Engineering*. Gulf Professional.

Nasteski, V., 2017. "An overview of the supervised machine learning methods". *Horizons. b*, Vol. 4, pp. 51–62.

Ongsulee, P., 2017. "Artificial intelligence, machine learning and deep learning". In *2017 15th International Conference on ICT and Knowledge Engineering (ICT&KE)*. IEEE, pp. 1–6.

Pacheco, C.A.R. and Pereira, N.S., 2018. "Deep learning conceitos e utilização nas diversas áreas do conhecimento". *Revista Ada Lovelace*, Vol. 2, pp. 34–49.

Pedersen, 2018. "Tensorflow tutorial 23 time-series prediction". `https://github.com/Hvass-Labs/TensorFlow-Tutorials/blob`
$Series - Prediction.ipynb. Accessed$: $2022 - 06 - 01$.

Rauber, T.W., 2005. "Redes neurais artificiais". *Universidade Federal do Espírito Santo*, Vol. 29.

Rodrigues, J.N., 2021. "Inflação no brasil: uma aplicação de séries temporais e redes neurais recorrentes".

Rusk, N., 2016. "Deep learning". *Nature Methods*, Vol. 13, No. 1, pp. 35–35.

Silva, M., 2010. *Aplicação de redes neurais e artificiais no diagnóstico de falhas de turbinas a gás*. Ph.D. thesis, Dissertação (Mestrado)—PUC-RIO-PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE ....

Tian, C., 2018. *Machine learning approaches for permanent downhole gauge data interpretation*. Stanford University.

Tian, C. and Horne, R.N., 2019. "Applying machine-learning techniques to interpret flow-rate, pressure, and temperature data from permanent downhole gauges". *SPE Reservoir Evaluation & Engineering*, Vol. 22, No. 02, pp. 386–401.

7. RESPONSIBILITY NOTICE

The author(s) is (are) solely responsible for the printed material included in this paper.