Neurocomputing 323 (2019) 203–213

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Time series forecasting of petroleum production using deep LSTM recurrent networks

Alaa Sagheer[a,b,*], Mostafa Kotb[b]

[a] College of Computer Science and Information Technology, King Faisal University, Saudi Arabia
[b] Center for Artificial Intelligence and Robotics (CAIRO), Faculty of Science, Aswan University, Egypt

ARTICLE INFO

Article history:
Received 11 November 2017
Revised 4 July 2018
Accepted 30 September 2018
Available online 5 October 2018

Communicated by Zidong Wang

Keywords:
Time series forecasting
Deep neural networks
Recurrent neural networks
Long-short term memory
Petroleum production forecasting

ABSTRACT

Time series forecasting (TSF) is the task of predicting future values of a given sequence using historical data. Recently, this task has attracted the attention of researchers in the area of machine learning to address the limitations of traditional forecasting methods, which are time-consuming and full of complexity. With the increasing availability of extensive amounts of historical data along with the need of performing accurate production forecasting, particularly a powerful forecasting technique infers the stochastic dependency between past and future values is highly needed. In this paper, we propose a deep learning approach capable to address the limitations of traditional forecasting approaches and show accurate predictions. The proposed approach is a deep long-short term memory (DLSTM) architecture, as an extension of the traditional recurrent neural network. Genetic algorithm is applied in order to optimally configure DLSTM's optimum architecture. For evaluation purpose, two case studies from the petroleum industry domain are carried out using the production data of two actual oilfields. Toward a fair evaluation, the performance of the proposed approach is compared with several standard methods, either statistical or soft computing. Using different measurement criteria, the empirical results show that the proposed DLSTM model outperforms other standard approaches.

1. Introduction

Time Series Forecasting (TSF) system involves predicting the system behavior in future, which is based on information of the current and past status of the system. Presently, TSF plays an imperative role in several real world problems, such as the financial markets, network traffic, weather forecasting, and petroleum (or oil) industry, among others [1]. In the past, the TSF problem has been influenced by linear statistical methods in order to achieve the forecasting activities. Recently, several useful nonlinear time series models were proposed such as the bilinear model [2], the threshold autoregressive model [3] and the Autoregressive Conditional Heteroscedastic (ARCH) model [4], among others. However, the analytical study of non-linear time series analysis is still in its infancy compared to linear time series [1].

In the last two decades, several Artificial Neural Network (ANN) algorithms have drawn attention and have established themselves as serious contenders to statistical methods in the forecasting community after they showed better prediction accuracies [5]. Given the several ANN algorithms, identifying a specific ANN algorithm for a forecasting task should be based on a compromise among three aspects; namely, the complexity of the solution, the desired prediction accuracy, and data characteristics [5]. Considering the first two aspects, i.e. precision and complexity, the best results are obtained by the Feed Forward NN predictor, in which the information goes through the network in the forward direction only. However, on the addition of the third aspect, i.e. the data characteristics, Recurrent Neural Network (RNN) is found to be more suitable than FFNN [6].

In RNN, the activations from each time step are stored in the internal state of the network in order to provide a temporal memory property [7]. However, the most major weakness of RNN is carried out during the requirement of learning long-range time dependencies [7,8]. To overcome this drawback, Hochreiter and Schmidhuber [9] developed the Long Short-Term Memory (LSTM) algorithm as an extension to RNN [8,10]. Despite the advantages cited for LSTM and its predecessor RNN, their performances for TSF problem are not satisfactory. Such shallow architectures, can not represent efficiently the complex features of time series data, particularly, when attempting to process highly nonlinear and long interval time series datasets [8,11].

* Corresponding author at: College of Computer Science and Information Technology, King Faisal University, Saudi Arabia And Faculty of Science, Aswan University, Egypt
E-mail addresses: asagheer@kfu.edu.sa, asagheer@aswu.edu.eg (A. Sagheer).

https://doi.org/10.1016/j.neucom.2018.09.082
0925-2312/

In this paper, we propose that a Deep LSTM (DLSTM) architecture can adapt with learning the nonlinearity and complexity of time-series data. The proposed deep model is correspondingly an extension of the original LSTM model, where it includes multiple LSTM layers such that each layer contains multiple cells. The proposed model demonstrates a more effective use of the parameters of each LSTM's layer in order to train the forecasting model efficiently. It works as follows: each LSTM layer operates at different time scale and, thereby, processes a certain part of the desired task and, subsequently, passes it on to the next layer until finally the last layer generates the output [12,13].

Thus, we can attribute the benefit of stacking more than LSTM layer to the recurrent connections between the units in the same layer, and the feed-forward connections between units in an LSTM layer and the LSTM layer above it [13,14]. This ensures an improved learning with more sophisticated conditional distributions of any time series data. Also, it can perform hierarchical processing on difficult temporal tasks, and more naturally, capture the structure of data sequences [11].

Towards fair evaluation, here we in this study train and validate the DLSTM model through more than a scenario, where we have used the genetic algorithm in order to optimally design and configure the best DLSTM architecture and parameters. Concurrently, we compare the DLSTM's performance with the performance of other reference models using the same datasets, and same experimental conditions via different error measures. The reference models vary from statistical methods, neural networks (shallow and deep) methods, and hybrid (statistical and neural networks) methods.

The remainder of the paper is organized as follows: Section 2 describes the TSF problem and associated works in the oil and petroleum industry. The proposed DLSTM model is presented in Section 3. Section 4 shows the experiment settings of this paper. The experimental results of two case studies are shown in Section 5. Discussion and analysis of the results are provided in Section 6 and, finally, the paper is concluded in Section 7.

2. TSF problem statement

The majority of real-world time series data sets have a temporal or time sequence property, particularly, in forecasting activities for weather, stock markets, robotics, and oilfields production, among others. Correspondingly, it has been observed that finding an effective method for forecasting trends in time-series datasets continues to be a long-standing unsolved problem with numerous potential applications [1]. For this reason, time series forecasting is considered as one of the top ten challenging problems in data mining due to its unique properties [15]. In this paper, we focus on the TSF problem of petroleum fields production.

2.1. Overview of petroleum TSF

Forecasting of the petroleum production is a very pertinent task in the petroleum industry, where the accurate estimation of petroleum reserves involves massive investment of money, time and technology in the context of a wide range of operating and maintenance scenarios [16,17]. As such, a fairly precise estimation of petroleum quantity in the reservoir is in high demand [17,18]. However, several characteristics of petroleum time series data make such estimations more challenging.

First of all, the samples of petroleum time series data often contain excessive noise, defects and anomalies, and, also sometimes, high dimensionality [19,20]. *Second*, the petroleum time series datasets are non-stationary and may exhibit variable trends by nature [21].[1] This implies that the statistical characteristics of the data, such as frequency, variance, and mean, undergo alternation over time [11]. *Third*, the rock and fluid properties of the reservoirs are highly nonlinear and heterogeneous in nature [19].

It is known that, the petroleum production from a reservoir is dependent on several dynamic parameters, such as fluid saturation and pressure in the reservoir, and static parameters such as porosity and permeability [18]. The majority of these parameters are not always available. Certainly, this limited data access from the petroleum reservoirs lessens the overall accuracy of forecasting [11].

2.2. Related works

Several approaches have been developed to overcome the aforementioned petroleum TSF challenges, however, yet the key for a successful forecasting lies in choosing the right representation among these approaches [11]. These approaches can be classified into two broad categories; namely, statistical approaches, and soft computing approaches. One of the most common traditional statistical methods is the Autoregressive Integrated Moving Average (ARIMA) [22].

ARIMA and its variants can be used to achieve diverse forecasting activities in the petroleum industry such as, prices, consumption levels, and reservoir production [23]. Another known mathematical method is the Decline Curve Analysis (DCA) method, which is based on the conventional ARPs equation. Historically, DCA has been widely used in petroleum industry, particularly in the scenarios depicting the decline of petroleum production with the increase in production time [24].

Nevertheless, the performance of traditional mathematical methods is still questionable. Indeed, more complex, high-dimensional, and noisy real-world time-series data cannot be described with analytical equations based on parameters, in order to solve since the dynamics that are either too complex or unknown [11], as the case of DCA. Moreover, the main drawback of traditional methods is that these methods are based mainly on the analysis of subjective data types. In other words, they pick the proper slope, and subsequently tune in the parameters of the numerical simulation model, in such a way that the reasonable values are retained, and finally, they are able to provide interpretations of the oilfield's geology [25]. But the oilfield's geology and fluid properties of the oilfields are highly nonlinear and heterogeneous in nature, thus yielding time series data that represent a long memory process. Certainly these properties represent big challenges for traditional approaches, which still are far from estimating the accurate future production of petroleum [11,17].

Since the past decade, sincere efforts have been evidently published in the literature presenting the use of soft computing methods to achieve different forecasting activities in a number of petroleum engineering applications. In 2011, Berneti and Shahbazian presented an imperialist competitive algorithm using ANN to predict oil flow rate of the oil wells [26]. In 2012, Liu et al. combined wavelets transformation with ANN in order to establish a production-predicting model that used drill stem to test production and wavelet coefficients [27]. In 2013, Chakra et al. presented an innovative higher-order NN model to focused on forecasting cumulative oil production from a petroleum reservoir located in Gujarat, India [18].

More recently in 2016, Aizenberg et al. presented a multilayer NN with multi-valued neurons capable of performing a time series forecasting of oil production [25]. Aizenberg model is based on a complex-valued neural network with a derivative-free backpropagation-learning algorithm. Eventually, Ma presented an extension of the Arps decline model, which was constructed within a nonlinear multivariate prediction approach [28]. The approach is considered as a hybrid approach that combines the kernel trick

[1] see chapter(3–5) in [21].

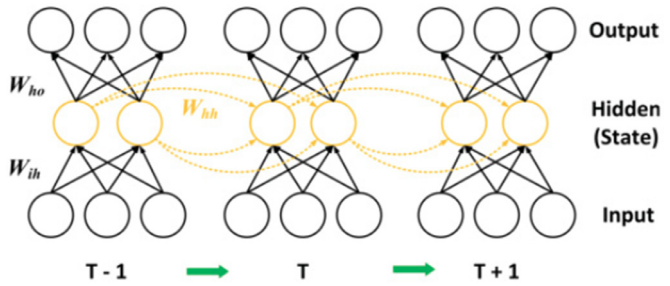

Fig. 1. Processing of time sequence in RNN.

with the Arps exponential decline equation. It is worthy to mention that, in this paper we conduct a comparison with both Chekra et al. [18] and Ma [28] approaches since both of these contributions present application of the same case studies as described in this paper.

2.3. Motivation

Although the soft computing methods, that employ different ANN algorithms, are used to recover the aforementioned limitations of the statistical methods and yield more accurate forecasting rates, they are observed to still face some challenges. It is demonstrated that the traditional ANNs with shallow architectures are devoid of sufficient capacity to accurately model the aforementioned complexity aspects of time series data, such as, high nonlinearity, longer intervals, and big heterogeneous properties [28,29].

This reason, and more, motivated us to solve the TSF problem using a Deep Neural Network (DNN) architecture instead of shallow NN architecture models. DNNs models are termed deep because they are constructed by stacking multiple layers of nonlinear operations one top of one another with several hidden layers [30].

3. The proposed model

Prior to introducing the proposed model, it is essential to describe briefly the original LSTM as in the current study it is the precedent of the proposed model.

3.1. The original LSTM model

The traditional Recurrent Neural Network (RNN), aka vanilla RNN, is one of the recursive neural network approaches that can be applied for modeling of sequential data. The key feature of RNN is the network delay recursion, which enables it to describe the dynamic performance of systems [6]. The signal delay recursion makes the output of the network at time t associate not only with the input at time t but also with recursive signals before time t, as shown in Fig. 1. However its capability to process short-term sequential data, the weakness of RNN is carried out when learning long-range dependencies, or long-term context memorization, is demanded in time series forecasting applications [8,10].

However, despite the introduction of several RNN variants, the Long Short-Term Memory (LSTM) model is the elegant RNN's variant, which uses the purpose-built LSTM's memory cell in order to represent the long-term dependencies in time series data [31]. In addition, LSTM is introduced to solve the vanishing gradient problem of RNN in case of a long-term context memorization is required [8,9]. The LSTM model, developed by Horchreiter and Schmidhuber [9], truncates the gradients in the network where it is innocuous by enforcing constant error flows through constant error carousels within special multiplicative units. These nonlinear units learn to open or close gates in the network in order to regulate such constant error flow [10]. Therefore, LSTM approximates the long-term information with significant delays by expediting the conventional RNN algorithm [31].

The key in LSTM structure is the cell state (memory cell), which looks like a conveyor belt. It runs straight down the entire chain with the ability to add or remove information to the cell state, carefully regulated by structures called gates. The gates are ways for optional inlet of information. They are composed of a sigmoid neural net layer and a pointwise multiplication operation as depicted in Fig. 2. An input at time step t is (X_t), and the hidden state from the previous time step (S_{t-1}) that is introduced to LSTM block, and then the hidden state (S_t) is computed as follows:

- The first step in LSTM is to decide what information is going to be thrown away from the cell state. This decision is made by the following forget gate (f_t):

$$f_t = \sigma\left(X_t U^f + S_{t-1} W^f + b_f\right) \tag{1}$$

- The following step is to decide which new information is going to be stored in the cell state. This step has two folds: First, the input gate (i_t) layer decides which values to be updated. Second, a tanh layer that creates a vector of new candidate values $\tilde{C}_t$. These two folds can be described as follows:

$$i_t = \sigma\left(X_t U^i + S_{t-1} W^i + b_i\right) \tag{2}$$

$$\tilde{C}_t = \tanh\left(X_t U^c + S_{t-1} W^c + b_c\right) \tag{3}$$

- Then, update the old cell state, C_{t-1} into the new cell state C_t, which can be given as:

$$C_t = C_{t-1} \otimes f_t \oplus i_t \otimes \tilde{C}_t \tag{4}$$

- Finally, decide what is going to be produced as output. This output will be based on the cell state, but will be a filtered version. In this step, the output gate (o_t) decides what parts of the cell state are going to be produced as output. Then, the cell state goes through tanh layer (to push the values to be between -1 and 1) and multiply it by the output gate as follows:

$$o_t = \sigma\left(X_t U^o + S_{t-1} W^o + b_o\right) \tag{5}$$

$$S_t = o_t \otimes \tanh(C_t) \tag{6}$$

From the previous six equations, the LSTM presents the following three groups of parameters:

1. *Input weights:* $\boldsymbol{U^f}$, $\boldsymbol{U^i}$, $\boldsymbol{U^o}$, $\boldsymbol{U^c}$.
2. *Recurrent weights:* $\boldsymbol{W^f}$, $\boldsymbol{W^i}$, $\boldsymbol{W^o}$, $\boldsymbol{W^c}$.
3. *Bias:* $\boldsymbol{b_f}$, $\boldsymbol{b_i}$, $\boldsymbol{b_o}$, $\boldsymbol{b_c}$.

3.2. The deep LSTM recurrent network

It is widely demonstrated that increasing the depth of a neural network is an effective way to improve the overall performance [30]. Encouraged by the impressive learning abilities of deep recurrent network architectures [32], we have developed a deep LSTM recurrent network to be used in time series forecasting applications. In the proposed DLSTM, we are able to stack several LSTM blocks, as shown in Fig. 3, one after another connected in a deep recurrent network fashion to combine the advantages of a single LSTM layer. The goal of stacking multiple LSTM in such a hierarchical architecture is to build the features at the lower layers that will disentangle the factors of variations in the input data and then combine these representations at the higher layers. In case of large or complex data, it is demonstrated that such deep architecture will generalize better due to a more compact representation than shallow architecture [11,14,32].

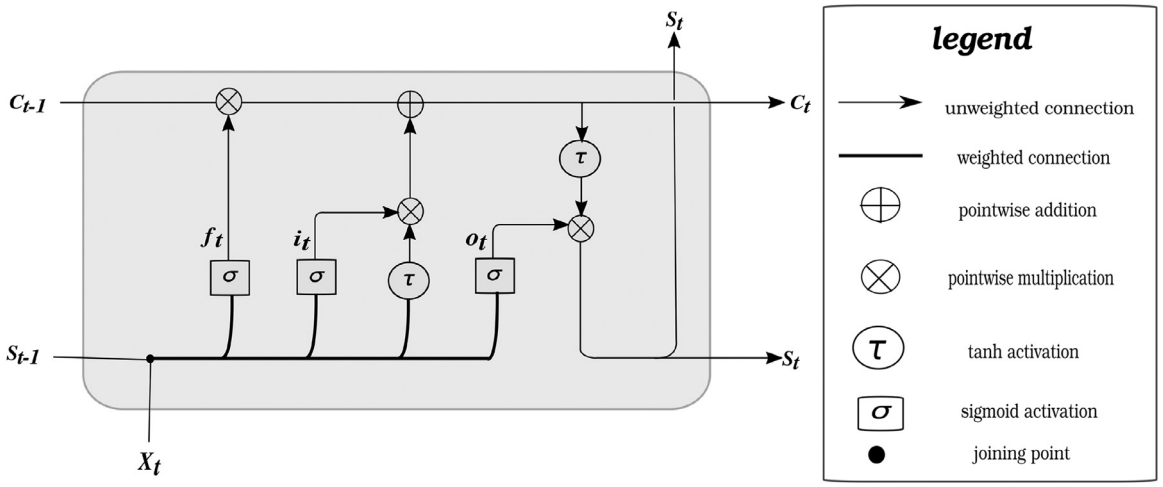

Fig. 2. LSTM block, where f_t, i_t, o_t are forget, input, and output gates respectively.

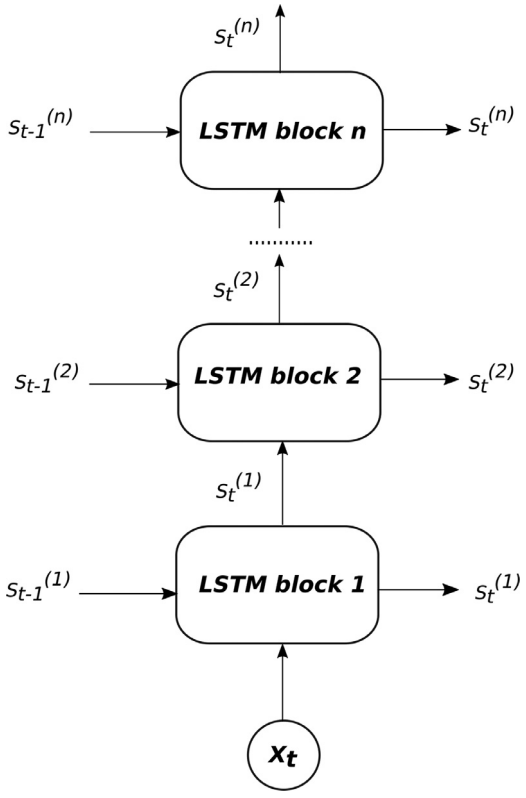

Fig. 3. The architecture of DLSTM recurrent network.

In the DLSTM architecture as shown in Fig. 3, the input at time t, X_t is introduced to the first LSTM block along with the previous hidden state $S_{t-1}^{(1)}$, the superscript (1) refers to the first LSTM. The hidden state at time t, $S_t^{(1)}$ is computed as shown in Section 3.1 and goes forward to the next time step and also goes upward to the second LSTM block. The second LSTM uses the hidden state $S_t^{(1)}$ along with the previous hidden state $S_{t-1}^{(2)}$ to compute $S_t^{(2)}$, that goes forward to the next time step and upward to the third LSTM block and so on, until the last LSTM block is compiled in the stack.

The benefit of such stacked architecture is that each layer can process some part of the desired task and subsequently pass it on to the next layer until finally the last accumulated layer provides the output. Another benefit, such architecture allows the hidden state at each level to operate at a different timescale. The last two benefits have great impact in scenarios showing the use of data with long-term dependency or in case of handling multivariate time series datasets [33].

4. Experiments

In this section, we will show in detail all the experimental settings that were adopted in order to implement both the proposed model and the reference models using real datasets. In addition, this section provides a brief overview of the reference models and the optimality criteria, we will rely on for a comparative analysis of their performance with the proposed model's performance. The codes that have been used in the experiments of this paper are shared in github.[2]

4.1. Experimental setting

The following experimental settings have been adopted for all experiments conducted in this paper.

4.1.1. Data preprocessing

The data used in this paper are the raw production data of two actual oilfields, so it is highly possible to include noise as in influencing factor. As such, it is not appropriate to use the raw production data in the learning of NN because NN requires extremely low learning rates. Thus, a preprocessing scenario consists of four steps has been incorporated before the use of the raw production data in the experiments of this paper.

Step 1: *Reduce noise from raw data*

To smoothen the raw data and remove any possible noise we will use the moving average filter as a type of low pass filter in the analogous way as described in [18]. Specifically, this filter provides a weighted average of past data points in the time series production data within a time span of five-points to generate smoothed estimation of a time series. This step is imperatively incorporated to reduce the random noise in data by retaining the sharpest step response associated with raw data [18].

[2] https://github.com/DeepWolf90/DLSTM2.

Step 2: *Transform raw data to stationary data*

As explained in Section 2.1, time series data are non-stationary data, which may, in fact, exhibit a specific trend [21]. Of course, stationary data is easier to model and will very likely result in more skillful forecasts. In the current preprocessing step, we removed the trend property in the data; whether it is increasing or decreasing trend. Later on, we added the trend back to forecasts in order to return the forecasting problem into the original scale and calculate a comparable error score. A standard way to remove the trend is by the differencing of the data. That is the observation from the previous time step *(t-1)* is subtracted from the current observation *(t)* [21].[3].

Step 3: *Transform data into supervised learning*

We use one-step ahead forecast, where the next time step *(t+1)* is predicted. We divide the time series into input (x) and output (y) using ***lag time method***, and specifically, in the study we have used different sizes of lag from **lag1** to **lag6**.

Step 4: *Transform data into the problem scale*

Like other neural networks, DLSTM expects data to be within the scale of the activation function used by the network. The default activation function for LSTM is the hyperbolic tangent (tanh), wherein its output values lie between −1 and 1. This is the preferred range for the time series data. Later on, we transformed the scaled data back in order to return the forecasting problem into the original scale.

4.1.2. Implementation scenario

The implementation experiments of the proposed DLSTM model include two different scenarios, namely, (i) *static* scenario and (ii) *dynamic* scenario. In the static scenario, we fit the forecasting model with all of the training data and then forecast each new time step once at a time with the testing data. In the dynamic scenario, we update the forecasting model each time step with the insertion of new observations from the testing data. In other words, the dynamic scenario uses the value of the previous forecasted value of the dependent variable to compute the next one, whereas the static forecast uses the actual value for each subsequent forecast.

4.1.3. Training of DLSTM

In the training phase of DLSTM experiments, we use the *Genetic Algorithm (GA)* to infer optimal selection for the proposed model hyper-parameters. We implemented the GA using *Distributed Evolutionary Algorithms in Python* (DEAP) library [34]. The number of hyper-parameters is based on the implementation scenario. For the *static scenario*, there are three hyper-parameters, namely,*number of epochs, number of hidden neurons, and the lag size*. For the *dynamic scenario*, there are four hyper-parameters, the same three hyper-parameters of the static scenario, plus the *number of updates*, which is the number of times we update our forecasting model each time step when new observations from the testing data are inserted. This methodology is typically adopted in the experiments of other neural networks in the reference models.

4.2. Reference models

Toward a fair evaluation, we will compare the proposed DLSTM model with different reference models that vary from statistical methods, machine learning methods, and hybrid (statistical and machine learning) methods. The reference models are:

1. The ARIMA model

The comparison with the Auto-Regression Integrated Moving Average (ARIMA) algorithm represents a statistical-based comparison. For comparison purpose, we will implement the ARIMA program using *Statsmodels* [35] library where we use the grid search to iteratively explore different combinations of the known ARIMA parameters *(p, d, q)*, see chapter 5 [21]. For each combination of these parameters, we fit a new ARIMA model and, subsequently, we use the *Akaike Information Criterion (AIC)* value to choose the best combination of these parameters [36]. The AIC measures how well a model fits with the data in consideration of the overall model complexity.

2. The Vanilla RNN model

The comparison with the vanilla RNN model represents a machine learning-based comparison. The original RNN is already covered briefly in Section 3.1. For comparison purpose, we will implement two RNN reference models, one with single hidden layer and the other one with multiple hidden layers [13,14].

3. The DGRU model

The comparison with the Deep Gated Recurrent Unit (DGRU) model represents a deep learning-based comparison, where DGRU is a counterpart of DLSTM. It is demonstrated that, the GRU model is similar to the original LSTM model with the exception that GRU includes only two gates rather than three [37]. The experiments of DGRU are typically similar to that of DLSTM.

4. The NEA model

The comparison with the Nonlinear Extension for linear Arps decline (NEA) model represents a hybrid-based comparison. NEA is a hybrid method that combines Decline Curve Analysis (DCA), which is a traditional statistical method, with the kernel machine, which is a machine learning method [28]. For comparison purpose, we will rely on the results provided in [28] using same datasets of both case studies described in this paper.

5. The HONN model

The comparison with the Higher-Order Neural Network (HONN) model is another machine learning-based comparison, where HONN is a feedforward multilayer neural network model that employs what is called higher-order synaptic operations (HOSO). HOSO of HONN embraces the linear correlation (conventional synaptic operation) as well as the higher-order correlation of neural inputs with synaptic weights [18]. For the comparison purpose, we will rely on the results introduced by the authors of [18] by exclusively using the second case study, since they did not apply their method on the first case study described in this paper.

4.3. Forecasting accuracy measures

In the literature, two kinds of errors are usually measured in order to estimate the forecasting precision and performance evaluation of the forecasts, namely, scale-dependent errors and percentage errors.

(1) Scale-dependent errors These errors are on the same scale as the data itself. Therefore, as a limitation, the accuracy measures that are based directly on this error cannot be used to make comparisons between series that are on different scales. The known scale-dependent measure is therefore based on the squared error, namely, root mean square error (RMSE) [38], which can be given as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i^{obs} - y_i^{pred})^2}, \tag{7}$$

Where, y_i^{obs} is the current observation and y_i^{pred} is its predicted value.

(2) Percentage errors Percentage errors have the advantage of being scale-independent, and therefore are frequently used to

[3] chapter 3 in [21].

Table 1
Best results of DLSTM with static scenario.

No. of layer	No. of hidden units	No.of Epochs	lag	RMSE	RMSPE
1	[4]	953	5	0.234	3.337
2	[4,2]	787	5	0.227	3.253
3	**[5,4,2]**	**800**	**5**	**0.209**	**2.995**

compare forecast performance between different scaled datasets. The most commonly used measure is the root mean square percentage error (RMSPE) [38], which can be given as follows:

$$RMSPE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left[\frac{y_i^{pred} - y_i^{obs}}{y_i^{obs}}\right]^2} \times 100 \quad (8)$$

It is clear that, both measures are calculated by comparing the target values for the time series and its corresponding time series predictions. The results obtained using both metrics are different in their calculated values, but the significance of each metric is similar in performance measurement of the prediction models. Notably, since the production data presents different scales in the majority of cases, it is preferable to rely on RMSPE, or any other percentage error measures, for estimating the relative error between different models [38].

5. Experimental results

We proceed now to show the quantitative and visual results of the proposed DLSTM model along with the reference models for each case study. Notably, the results shown in all tables of this section indicate the performance of the corresponding model in the testing data rather than training data. This has been done in concurrence with the widely demonstrated fact, which states, the genuine evaluation for forecasting performance should be based on unseen data not the historical (training) data, which already seen by the model [39].[4]

5.1. Case study 1: Using production data of Block-1 of Huabei oil field in China

This case study includes raw data collected from the Block-1 in Huabei oilfield, which is located in north China [28].[5] The dataset of this oilfield contains 227 observations of the oil production data, in which the first 182 observations (80% of dataset) have been used to build, or train, the forecasting models, and the remaining 45 observations (20% of the dataset) have been used for testing the performance of the forecasting models.

The best performance results of the proposed DLSTM static scenario, DLSTM dynamic scenario, single-RNN, Multi-RNN, and DGRU are shown separately in Tables 1–5, respectively. Each of these five tables show the values of each hyper-parameter, which has been optimally selected using the GA as described in Section 4.1.3. The relation between the original production data and their prediction for the DLSTM model is illustrated in Figs. 4 and 5. Table 6 shows an overall comparison among these five models along with the best parameter combinations of ARIMA method and the best performance results of NEA model reported in [28][6] using same data set. The NEA results shown in Table 6 are imparted as they are given by the authors of [28] where they did not consider the RMSE measure.

[4] See section 3.4 , pages 177:184.
[5] The raw data are listed in the Table 1 in [28].
[6] Relation between the original production and the prediction results of NEA is plotted in Fig. 1 [28].

5.2. Case study 2: Using production data of Cambay Basin oil field in India

As the previous case study, we examined the proposed model and the reference models using real production data collected through six years from 2004 to 2009, i.e. about 63 months. This oilfield is located in the southwestern part of Tarapur Block of Cambay Basin to the west of Cambay Gas Field in India [18].[7] This oilfield consists of total eight oil producing wells that present continuous production history. The authors in [18,28] considered only the cumulative oil production data from five wells; out of these eight wells. Thus implying the availability of five input series corresponding to the monthly production of the five oil wells, plus an output series as corresponding to the cumulative production of this oilfield. The relationship between the five input series and the output series has been reported to be highly nonlinear [18].

Accordingly, and toward fair evaluation, in the experiments of this case study we will consider also the same cumulative data of the same five wells. We will follow the same experimental scenario described in [18,28] by dividing the production dataset into two sets, i.e. first set (70% of data set) to be used to build the forecasting models, and second set (30% of the data set) to be used for testing the performance of the forecasting models. The results of each model shown in this section are based on the testing data.

The best performance results of the proposed DLSTM static scenario, DLSTM dynamic scenario, single-RNN, Multi-RNN, and DGRU are shown separately inTables 7–11, respectively. Each table of these five tables shows the values of each hyper-parameter, which optimally selected using the GA as described in Section 4.1.3. The relation between the original production data and their prediction for the DLSTM model is illustrated in Figs. 6 and 7. Table 12 shows an overall comparison among these five models along with the best parameter combinations of ARIMA method and the best performance results of NEA reported in [28] using the same data set. The NEA results shown in Table 12 are imparted as they are given by the authors of [28] where they did not consider the RMSE measure.

This case study provides an extra comparison where we compare the proposed DLSTM model with the HONN model [18], described in Section 4.2. In their paper, the authors used three measures to evaluate their model and these include *MSE, RMSE, and MAPE*. In the current paper, we have used the RMSE (RMSE is the root of MSE) as described in Section 4.3. Subsequently, in this comparison we calculate the MAPE measure within our model to compare with the MAPE results of HONN shown in [18]. The MAPE, as a percentage error measure, can be computed as follows:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{|y_i^{pred} - y_i^{obs}|}{y_i^{obs}}\right] \times 100 \quad (9)$$

Table 13 shows the comparison between the HONN model and the proposed DLSTM model based on the three measures. For the proposed DLSTM model, the best results of both scenarios (static and dynamic) are shown in Table 13. The authors of [18] used three different lags in their experiments, and the best result as highlighted by them was inferred using *lag 1* [18][8] which is included in Table 13.

6. Results analysis and discussion

In this paper, we tried to ensure a genuine evaluation for the proposed model against five different types of comparison with state-of-the-art techniques using two real world datasets. More

[7] The raw data are listed in the Table 1b in [18].
[8] see Table 3 in [18].

Table 2
Best results of DLSTM with dynamic scenario.

No. of layer	No. of hidden units	No.of Epochs	lag	update	RMSE	RMSPE
1	[3]	1352	3	1	0.267	3.783
2	**[4,5]**	**1187**	**5**	**1**	**0.219**	**3.124**
3	[4,3,3]	403	5	2	0.257	3.637

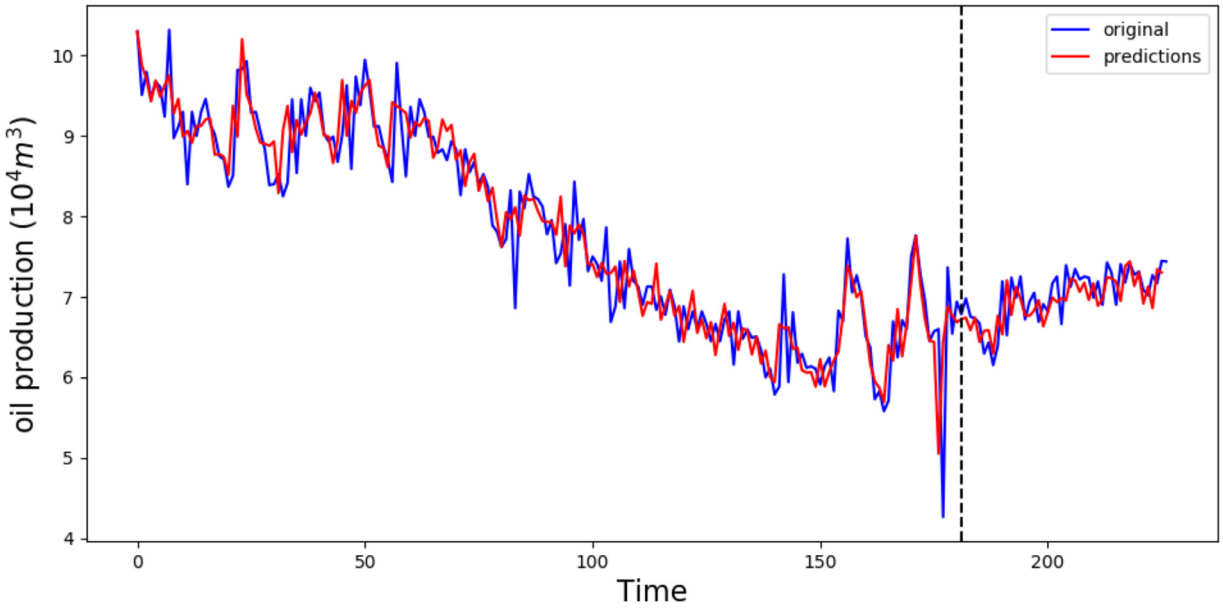

Fig. 4. Production data v.s. prediction using DLSTM-static.

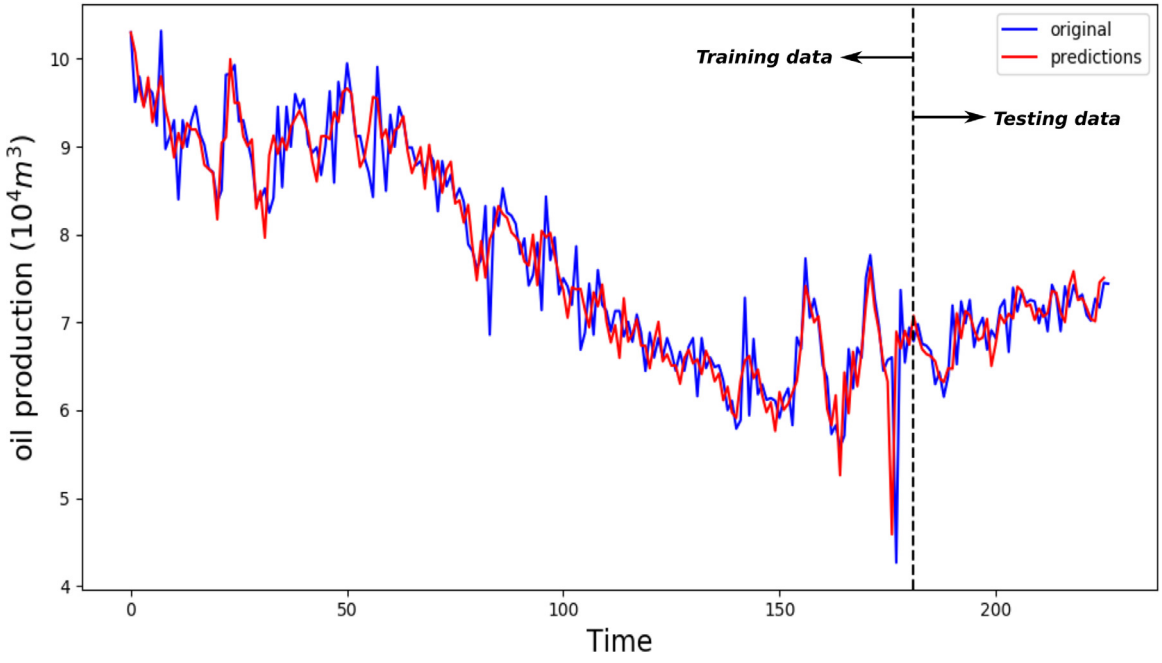

Fig. 5. Production data v.s. prediction using DLSTM-dynamic.

than one standard optimality criteria are used to assess the performance of each model. It is widely demonstrated in literature that the percentage error measures are the most appropriate tool to assess the performance of different forecasting models. It also presents the percentage error capable to estimate the relative error between different models particularly when the samples of the time series data have different scales [39]. Accordingly, in this section we will discuss and analyze the results shown in the previous section where we will focus on these results based on the percentage error measure of each model.

6.1. Case 1 versus Case 2

However it is not a real comparison since each case study has its own samples and source, but we can notice few observations on

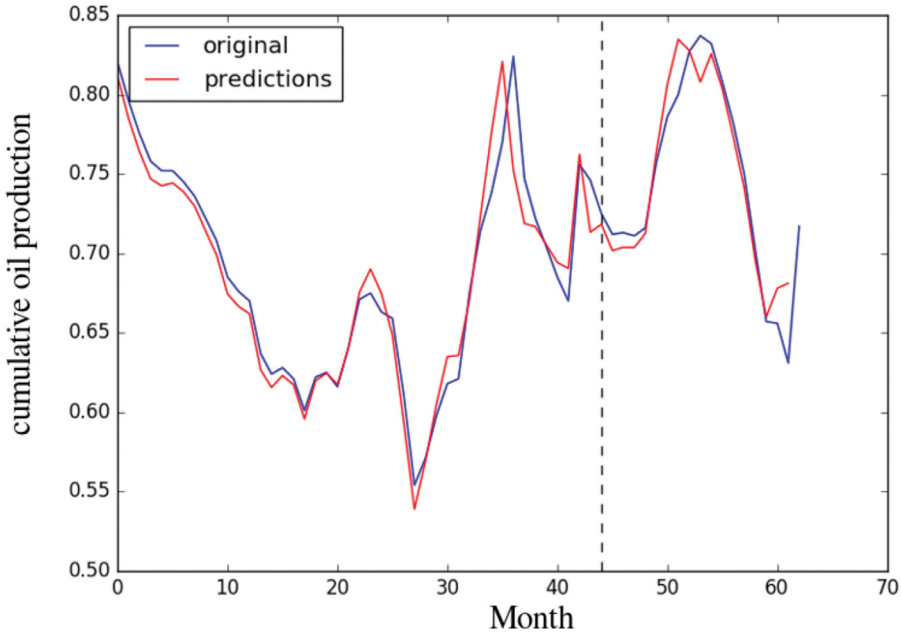

Fig. 6. Production data v.s. prediction using DLSTM-static.

Table 3
Best results of Single-RNN.

No. of units	No.of Epochs	lag	RMSE	RMSPE
[4]	**1890**	**5**	**0.233**	**3.290**
[5]	653	4	0.238	3.366
[3]	431	4	0.263	3.740

Table 4
Best results of Multi-RNN.

No. of layer	No. of hidden units	No.of Epochs	lag	RMSE	RMSPE
2	**[2,4]**	**1551**	**5**	**0.219**	**3.129**
2	[3,4]	1913	5	0.239	3.387
2	[2,2]	787	5	0.247	3.530
3	[5,5,4]	457	3	0.258	3.701
3	[4,3,4]	1611	5	0.237	3.374

Table 5
Best results of DGRU.

No. of layer	No. of hidden units	No.of Epochs	lag	RMSE	RMSPE
1	[4]	1870	2	0.256	3.610
2	[4,3]	1011	6	0.237	3.391
2	**[5,3]**	**1514**	**6**	**0.222**	**3.175**
3	[4,3,1]	354	6	0.263	3.734

Table 6
Overall comparison among ARIMA, NEA [28], RNN, DGRU, and DLSTM using data set of case study 1.

Forecasting model	RMSE	RMSPE
ARIMA	0.310	4.705
NEA [28]	–	4.221
DLSTM(static)	**0.209**	**2.995**
DLSTM(dynamic)	0.219	3.124
Single-RNN	0.233	3.290
Multi-RNN	0.219	3.129
DGRU	0.222	3.175

Table 7
Best results of DLSTM with static scenario.

No. of layer	No. of hidden units	No.of Epochs	lag	RMSE	RMSPE
1	**[3]**	**1700**	**3**	**0.025**	**3.496**
2	[1,1]	2000	1	0.030	4.135
3	[2,2,1]	2000	2	0.028	3.926

both case studies. In case study 1, we can notice from Tables 1 and 2 that the best results of DLSTM are achieved using three LSTM layers and two LSTM layers in static and dynamic scenario, respectively. Also, in Tables 4 and 5 we can notice that the best results of multi-RNN and DGRU are achieved using two layers in both cases. From the accumulated comparison in Table 6, it is clear that the values of the proposed DLSTM model are the global minimum values amongst the other reference models. In Tables 7, 8 and 10–13 of the other case study, we will notice the same pattern for all models, again, with a superiority for the DLSTM over the other models.

However, the DLSTM is the optimum among the other counterparts, though it illustrates a light variation in the hyper-parameters values, particularly the parameter “number of layers”. In our opinion, this variation in the best hyper-parameter values between the

Table 8
Best results of DLSTM with dynamic scenario.

No. of layer	No. of hidden units	No.of Epochs	Lag	Update	RMSE	RMSPE
1	[5]	1259	6	4	0.029	4.219
2	**[2,5]**	**1500**	**6**	**3**	**0.028**	**4.060**
3	[4,4,5]	1400	6	4	0.032	4.482

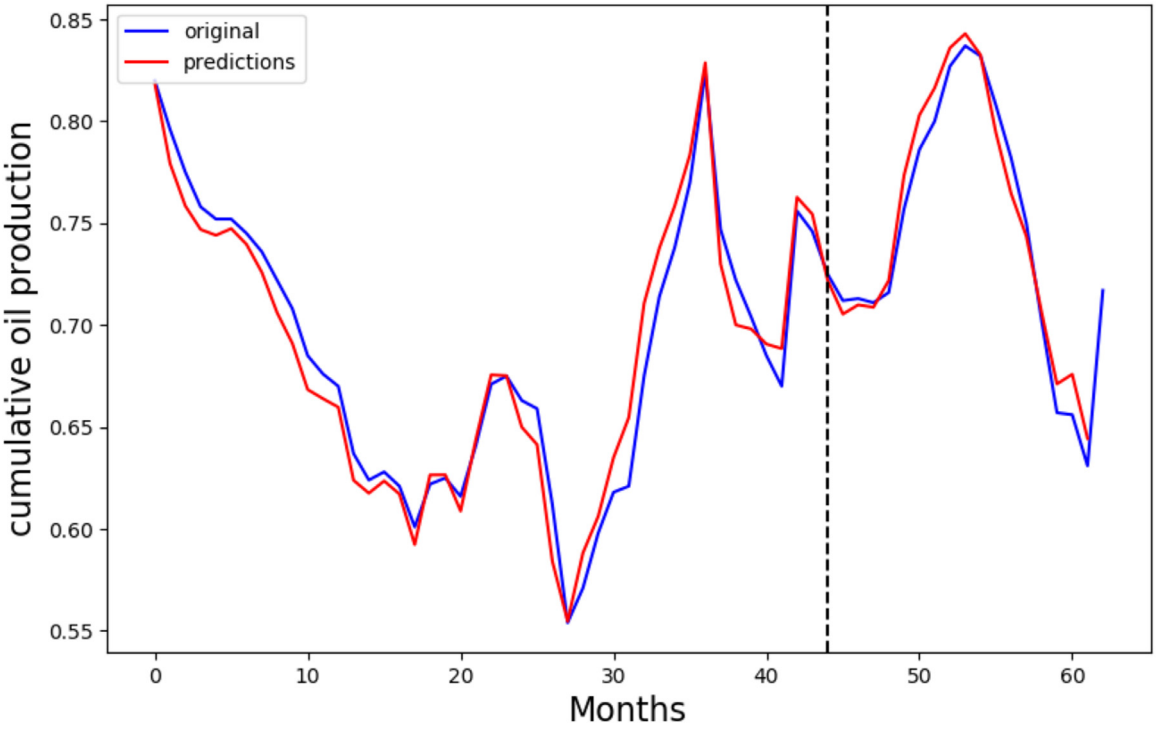

Fig. 7. Production data v.s. prediction using DLSTM-dynamic.

Table 9
Best results of Single-RNN.

No. of units	No.of Epochs	lag	RMSE	RMSPE
[1]	**1551**	**4**	**0.029**	**4.095**
[2]	1115	1	0.029	4.133
[1]	953	2	0.030	4.174

Table 10
Best results of Multi-RNN.

No. of layer	No. of hidden units	No.of Epochs	lag	RMSE	RMSPE
2	**[5,1]**	**1514**	**5**	**0.027**	**3.731**
2	[2,4]	1551	5	0.028	4.125
2	[2,2]	787	3	0.030	4.196
3	[1,1,3]	953	4	0.029	4.112
3	[1,3,3]	953	2	0.031	4.353

two case studies may be attributed to the higher data samples in case study 1 than case study 2. In other words, DLSTM does not require large number of layers in case the dataset size is not large. Of course, as the number of data samples is going to be bigger, essentially the performance of DLSTM going to be better [32].

6.2. DLSTM versus ARIMA

It is easy to notice in Table 6 for case study 1, where the dataset is large, all the errors of DLSTM are smaller than those of the ARIMA algorithm. Specifically, ARIMA achieved 4.7 minimum error whereas DLSTM achieved 2.9. The same pattern for case study 2 is presented in Table 12, where ARIMA achieved 3.8 minimum error whereas DLSTM achieved 3.5. In other words, the DLSTM model shows more efficiency than ARIMA model in predicting the future oil productions and in describing the typical tendency of the oil production as shown in Fig. 4–7. In contrast, the predicted values by ARIMA are quite far away from the oil production points where

Table 12
Overall comparison among ARIMA, NEA [28], RNN, DGRU, and DLSTM using data set of case study 2.

Forecasting Model	RMSE	RMSPE
ARIMA	0.027	3.773
NEA [28]	–	4.221
DLSTM(static)	**0.025**	**3.496**
DLSTM(dynamic)	0.028	4.060
Single-RNN	0.029	4.095
Multi-RNN	0.027	3.731
DGRU	0.028	3.991

Table 13
Comparison between HONN[18] and DLSTM.

Forecasting model	MSE	RMSE	MAPE
HONN[18]	0.001	0.035	3.459
DLSTM(static)	**0.000**	**0.025**	**2.851**
DLSTM(dynamic)	0.000	0.028	2.976

Table 11
Best results of DGRU.

No. of layer	No. of hidden units in each layer	No.of Epochs	lag	RMSE	RMSPE
1	[2]	787	2	0.029	4.125
2	[3,1]	431	4	0.030	4.207
3	[1,3,5]	354	6	0.029	4.035
3	**[4,3,5]**	**354**	**6**	**0.028**	**3.991**

the difference between both contenders approaches 2 points in first case. We can estimate why the performance of ARIMA is not well due to its linearity nature whereas the relationship between inputs and outputs is not linear in such a production data. As a nonlinear model, DLSTM could to describe smoothly the nonlinear relationship between inputs and outputs.

6.3. DLSTM versus other recurrent NNs

In this comparison, DLSTM is compared with its forefather, RNN, and its counterpart, DGRU, where the three contenders have the same origin and classified as recurrent neural networks. It is easy to notice in Table 6 for case study 1 that DLSTM achieved 3.4 against 3.7 for Multi-RNN and 4.0 for DGRU. The same rates are approximately achieved in case study 2 and Table 12. However, the error differences are not so big among the three contenders, since all of them have typical deep architecture, but still the proposed DLSTM model shows better performance than the others. Of course, as the size of data is going to be large, expressively the performance of DLSTM will be much better than RNN but may be similar to DGRU.

6.4. DLSTM versus reported approaches

This is the most important comparison between the proposed DLSTM model and other reported approaches, the NEA model [28] and the HONN model [18] since these three models are nonlinear and present different origins. For the NEA model, it is clear in Tables 6 12 that the DLSTM model outperforms the NEA model with a difference approaches to one point in case study 1. Namely, DLSTM achieved 2.9 against 4.2 achieved by NEA, whereas in case study 2, the DLSTM achieved 3.4 against 4.2 achieved by NEA. This indicates that the DLSTM model is more accurate than the NEA model in predicting the future oil production.

Superiority in performance is not the only advantage of DLSTM over NEA but also NEA performance is evidenced to be highly dependent on the selection of several parameters, as explained by the authors of [28]. Among these parameters, the most important parameters, which may affect the NEA performance includes:(i) the regularized parameter (γ), which controls the smoothness of the model, and (ii) the kernel parameter (σ) of the Gaussian kernel used in the NEA model. It is demonstrated by the authors in [28] that the NEA's performance is sensitive to the values of these two parameters. Accordingly, to investigate the performance of NEA model in the prediction of oil production, several experiments should be conducted in order to find improved and suitable combinations of these parameters.

Furthermore, the performance of these parameters in training phase is totally reversed in testing phase. For example, the training errors are growing with larger (σ) , whereas it is decreased for testing errors. The converse will be in the case of (γ) parameter, where training errors decrease with larger (γ) but the testing errors remain monotonic. If the designer is not aware of this relationship, the larger values of (σ) will convert the model from nonlinear behavior to linear behavior [28]. In other words, the overall performance of NEA in training phase and testing phase is not sufficiently harmonious and require careful deliberation with the parameters selections.

For the HONN model [18], we should highlight that this model is similar to traditional multilayer feed forward neural network. The difference here is that HONN employs what is called Higher-Order Synaptic Operations (HOSO). HOSO of HONN embraces the linear correlation (conventional synaptic operation) as well as the higher-order correlation of neural inputs with synaptic weights. In the paper of [18], different HOSO have been applied up to third-order, where the first-order, the second-order, and the third-order synaptic operations are called Linear Synaptic Operation (LSO), Quadratic Synaptic Operation (QSO) and Cubic Synaptic Operation (CSO), respectively [18]. The authors stated that the best HOSO operation is the third one (CSO).

It seems that the computation of HONN is complex where calculation of the activation function of the model is a combination of the conventional linear synaptic function plus the cubic synaptic operation. In addition, most of parameters, such as time lag and number of neurons in the hidden layer, are adjusted manually or based on trial and error. This means that the parameters selection should be adjusted carefully to ensure accurate oil production forecasting.

Nevertheless, in Table 13 DLSTM continues to show better performance than HONN via the three error measures, particularly for percentage error measure. Namely, through the MAPE measure the DLSTM achieved 2.8 against 3.4 for HONN. In our perspective, the optimality of DLSTM's performance can attribute to the recursive nature of DLSTM, against the feedforward nature of HONN. Indeed, the recursive property ensures more accurate prediction particularly when the dataset size going to be large.

7. Conclusion

In this paper, we developed a promising prediction model can be used in the majority of time series forecasting problems. However, in this paper, it is tested specifically in case of petroleum time series applications. The proposed model is a deep architecture of the Long-Short Term Memory (LSTM) recurrent network, where we denoted it as DLSTM. The paper empirically evidences that, stacking of more LSTM layers ensures to recover the limitations of shallow neural network architectures, particularly, when long interval time series datasets are used. In addition, the proposed deep model can describe the nonlinear relationship between the system inputs and outputs, particularly, if we knew that the petroleum time series data are heterogeneous and full of complexity and missing parts.

Notably, in the two case studies described in this paper the proposed model outperformed its counterparts deep RNN and deep GRU. In addition, the performance of the proposed DLSTM is observed to be much better than the statistical ARIMA model. The most important comparisons that conducted with two recent reported machine learning approaches, denoted as NEA and HONN, where DLSTM outperformed both of them with a noticeable difference on the scale of two different percentage error measures.

The accurate prediction and learning performance shown in the paper indicate that the proposed deep LSTM model, and other deep neural network models, are eligible to be applied in the nonlinear forecasting problems in the petroleum industry. In our future research plans, we will investigate the performance of DLSTM in other forecasting problems especially when the problem includes multi-variables (multivariate) time series data.

Acknowledgments

The authors of this paper would like to express about their thank and gratitude to “Deanship of Scientific Research” at King Faisal University, Saudi Arabia for their moral and financial support to this work under the research grant number (170069).

References

[1] J.G. De Gooijer, R.J. Hyndman, 25 years of time series forecasting, Int. J. Forecast. 22 (3) (2006) 443–473.
[2] D.S. Poskitt, A.R. Tremayne, The selection and use of linear and bilinear time series models, Int. J. Forecast. 2 (1) (1986) 101–114.
[3] H. Tong, Non-Linear Time Series: A Dynamical System Approach, Oxford University Press, 1990.

[4] R.F. Engle, Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation, Econometrica 50 (4) (1982) 987–1007.
[5] G. Zhang, B.E. Patuwo, M.Y. Hu, Forecasting with artificial neural networks: the state of the art, Int. J. Forecast. 14 (1998) 35–62.
[6] M.H. sken, P. Stagge, Recurrent neural networks for time series classification, Neurocomputing 50 (2003) 223–235.
[7] Bayer, J. Simon, Learning Sequence Representations, Technische Universitt Mnchen, 2015.
[8] Pascanu, Razvan, T. Mikolov, Y. Bengio, On the difficulty of training recurrent neural networks, in: Proceedings of the 30th International Conference on Machine Learning (3), volume 28, 2013, pp. 1310–1318.
[9] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[10] I. Sutskever, Training recurrent neural networks, University of Toronto, 2012 Ph.d. thesis.
[11] M. Längkvist, L. Karlsson, A. Loutfi, A review of unsupervised feature learning and deep learning for time-series modeling, Pattern Recognit. Lett. 42 (2014) 11–24.
[12] M. Hermans, B. Schrauwen, Training and analyzing deep recurrent neural networks, in: Proceedings of the 26th International Conference on Neural Information Processing Systems NIPS 1, 2013, pp. 190–198.
[13] R. Pascanu, C. Gulcehre, K. Cho, Y. Bengio, How to construct deep recurrent neural networks, in: Proceedings of the Second International Conference on Learning Representations ICLR, 2014.
[14] P.E. Utgoff, D.J. Stracuzzi, Many-layered learning, Neural Comput. 14 (10) (2002) 2497–2529.
[15] Q. Yang, X. Wu, 10 challenging problems in data mining research, Int. J. Inf. Technol. Decis. Making 5 (2006) 597–604.
[16] R. Mehrotra, R. Gopalan, Factors infleuencing strategic decision-making process for the oil/gas industriesof UAE-a study, Int. J. Mark. Financ. Manag. 5 (2017) 62–69.
[17] R.B.C. Gharbi, G.A. Mansoori, An introduction to artificial intelligence applications in petroleum exploration and production, J. Pet. Sci. Eng. 49 (2005) 93–96.
[18] N.C. Chakra, K.-Y. Song, M.M. Gupta, D.N. Saraf, An innovative neural forecast of cumulative oil production from a petroleum reservoir employing higher-order neural networks (HONNs), J. Pet. Sci. Eng. 106 (2013) 18–33.
[19] R. Nyboe, Fault detection and other time series opportunities in the petroleum industry, Neurocomputing 73 (10–12) (2010) 1987–1992.
[20] L. Martí, N. Sanchez-Pi, J. Molina, A. Garcia, Anomaly detection based on sensor data in petroleum industry applications, Sensors 15 (2015) 2774–2797.
[21] J.D. Cryer, K.-S. Chan, Time Series Analysis, 2nd ed., Springer Texts in Statistics, Springer, New York, 2008.
[22] S.L. Ho, M. Xie, The use of ARIMA models for reliability forecasting and analysis, Comput. Ind. Eng. 35 (1998) 213–216.
[23] J. Choi, D.C. Roberts, E. Lee, Forecasting oil production in north dakota using the seasonal autoregressive integrated moving average (s-ARIMA), Nat. Resour. 6 (2015) 16–26.
[24] A. Kamari, A.H. Mohammadi, M. Lee, A. Bahadori, Decline curve based models for predicting natural gas well performance, Petroleum 3 (2017) 242–248.
[25] I. Aizenberg, L. Sheremetov, L. Villa-Vargas, J. Martinez-Muñoz, Multilayer neural network with multi-valued neurons in time series forecasting of oil production, Neurocomputing 175 (2016) 980–989.
[26] S. Berneti, M. Shahbazian, An imperialist competitive algorithm artificial neural network method to predict oil flow rate of the wells, Int. J. Comput. Appl. 26 (2011) 47–50.
[27] Z. Liu, Z. Wang, C. Wang, Predicting reservoir production based on wavelet analysis-neural network, advances in computer science and information engineering, Adv. Intell. Soft Comput. 168 (2012).
[28] X. Ma, Predicting the oil production using the novel multivariate nonlinear model based on Arps decline model and kernel method, Neural Comput. Appl. 29 (2016) 1–13.
[29] S.B. Taieb, G. Bontempi, A.F. Atiya, A. Sorjamaa, A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition, Expert Syst. Appl. 39 (2012) 7067–7083.
[30] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015) 436–444.
[31] K. Greff, R.K. Srivastava, J. Koutnk, B.R. Steunebrink, J. Schmidhuber, LSTM: A search space odyssey, IEEE Trans. Neural Netw. Learn. Syst. 28 (2017) 2222–2232.
[32] M. Hermans, B. Schrauwen, Training and analysing deep recurrent neural networks, in: Proceedings of the 26th International Conference on Neural Information Processing Systems NIPS 1, 2013, pp. 190–198.
[33] S. Spiegel, J. Gaebler, A.L.E. De Luca, S. Albayrak, Pattern recognition and classification for multivariate time series, in: Proceeding of the Fifth International Workshop on Knowledge Discovery from Sensor Data, 2011, pp. 34–42.
[34] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, C. Gagné, DEAP: evolutionary algorithms made easy, J. Mach. Learn. Res. 13 (2012) 2171–2175.
[35] Seabold, Skipper, Perktold, Josef, Statsmodels: Econometric and statistical modeling with python, in: Proceeding of the 9th Python in Science Conference, 2010.
[36] E.J. Bedrick, C.L. Tsai, Model selection for multivariate regression in small samples, Biometrics 50 (1994) 226–231.
[37] C. Junyoung, C. Gulcehre, C. KyungHyun, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, in: Proceeding of the Deep Learning workshop at NIPS, 2014.
[38] R.J. Hyndman, A.B. Koehler, Another look at measures of forecast accuracy, Int. J. Forecast. 22 (4) (2006) 679–688.
[39] R.J. Hyndman, Measuring forecast accuracy, in: M. Gilliland, L. Tashman, U. Sglavo (Eds.), Business Forecasting: Practical Problems and Solutions, John Wiley & Sons, 2016, pp. 177–183.

Dr. Alaa Sagheer received his B.Sc. and M.Sc. in Mathematics from Aswan University, EGYPT. He got Ph.D. in Computer Engineering in the area of Intelligent Systems from the Graduate School of Information Science and Electrical Engineering, Kyushu University, Japan in 2007. After receiving PhD, he served as an Assistant Professor at Aswan University. In 2010, Dr. Sagheer established and directed the Center for Artificial Intelligence and Robotics (CAIRO) at Aswan University. He served also as the Principal Investigator in CAIRO in several research and academic projects funded by different Egyptian governmental organizations. In 2013, Dr. Sagheer and his team won the first prize, in a programming competition organized by the Ministry of Communication and Information Technology (MCIT) Egypt, for their system entitled Mute and Hearing Impaired Education via an Intelligent Lip Reading System. In 2014, he appointed as an Associate Professor at Aswan University. In the same year, Dr. Sagheer joined the Department of Computer Science, College of Computer Sciences and Information Technology, King Faisal University, Saudi Arabia. Dr. Sagheers research interests include artificial intelligence, machine learning, pattern recognition, computer vision, and optimization theory. Recently, Dr. Sagheer extended his research interest to include quantum computing and quantum communication. He authored and co-authored more than 40 research articles in his research interest fields. Dr. Sagheer is a member of IEEE and IEEE Computational Intelligence society. He is a reviewer for some journals and conferences related to his research interests.

Mostafa Kotb received his B.S. degree in computer science from Aswan University, Aswan, Egypt at 2012. He is now a master student and a research assistant in Center for Artificial Intelligence and Robotics (CAIRO), Aswan University. His research interests include artificial intelligence, machine learning, and deep learning.