SPE-196011-MS

Deep Learning for Well Data History Analysis

Yuanjun Li, Ruixiao Sun, and Roland Horne, Stanford University

This paper was prepared for presentation at the SPE Annual Technical Conference and Exhibition held in Calgary, Alberta, Canada, 30 Sep - 2 October 2019.

Abstract

The rapid development of machine learning algorithms and the massive accumulation of well data from continuous monitoring has enabled new applications in the oil and gas industries. Data gathered from well sensors are a foundation of the oilfield digitization and data-driven analysis. Here, we describe a deep learning approach to predict the long-term well performance based on a moderate duration of well monitoring data.

In this study, we first developed the data processing procedures for oilfield time series data and determined the proper selection of data sampling frequency, parameter combinations and data structures for deep learning models. Then we explored how Deep Learning (DL) models can be employed for well data analysis and how can we combine physics and DL models. Recurrent Neural Network (RNN) is a type of sequential DL model, which can be utilized for time series data analysis. This approach preserves preceding information and yields current response with memory of prior well behavior. Two candidate RNN models were tried to determine how well they were able to improve the accuracy and stability of well performance estimates. These two methods are Gated Recurrent Unit (GRU) and Long Short Term Memory (LSTM). In addition, a novel combination of RNN with Convolutional Neural Networks (CNNs), Long- and Short-term Time-series network (LSTNet), was also investigated.

These various models were tested and compared based on the public production datasets from Volve Field. Both GRU and LSTM achieved higher accuracy in performance prediction compared to the simple RNN. In the case of frequent well shut-in and opening, the failure in capturing fast pressure responses and the extreme fluctuations with the simple RNN ultimately leads to high error. In contrast, LSTNet is more stable to frequent or significant well variations. With advanced deep learning structures, engineers can interpret long-term reservoir performance information from responses estimated by deep learning models, instead of performing costly well tests or shut-ins.

Introduction

In the oil and gas industry, conventionally comprehensive reservoir information is required to extract physical models to construct the relationship between pressure and flow rate. Many ideal assumptions can make the simplified equations unrepresentative of the complex subsurface fluid behaviors. Another limitation for traditional modeling is that pressure prediction relies significantly on the flow rate as input.

However, from our observation, pressure is also strongly related to other time series data such as temperature and water cut. In addition, we may have other useful information measured at the same time as when we recorded the flow rate. Thus, we utilized deep learning models to help us to establish a more comprehensive system for the target reservoir by thoroughly utilizing all available parameters and depending less on the unreliable (and often missing) flow rate data. The characteristics of neural networks was also found to enable fast examination of noisy data.

Machine Learning for Well Data Analysis

Artificial neural network (ANN) is a popular and successful Machine Learning (ML) approach. Researchers started incorporating ANN for well data analysis from late last century. Those applications included feedforward neural network for reservoir characterization (An and Moon, 1993) and neural network for deconvolution (Essenreiter and Karrenbach, 1996). A recent work by Li et al. (2018) leveraged an ANN structure to estimate the stabilized reservoir pressure from early fall-off data of injection wells. However, a crucial limitation of the feedforward ANN architecture used in those studies is that prior calculations will not be taken into consideration for the next step.

Deep Learning for Well Data Analysis

Deep neural networks (DNN), which is a specific kind of ANN, is a hierarchical structure consisting more hidden layers. The fundamentals of deep learning concepts were first established in the mid twentieth century and improved subsequently. The rapid progress in computational hardware, such as the development of Graphics Processing Unit (GPU) and Tensor Processing Unit (TPU), enabled the intensive computational requirement and new breakthroughs.

Deep Learning methods have been studied intensively in many realms, including health care, automatic driving, advertising, robotics etc. The techniques have made extraordinary impacts on the solutions of a broad range of problems. However, researches on deep learning for well data history analysis is limited. Tian and Horne (2018) investigated the application of simple RNN and NARX for flow rate reconstruction. In this study, we explored other advanced deep learning structures for well data analysis.

Methodologies

Time series well production data contain pressure, multiphase flow rates and rate-related parameters such as water cut and choke opening percentage, temperature measured at wellhead and bottomhole, and other available measurements. The production data need to be sampled at appropriate frequency to enable transient analysis. Then a training data set needs to be created from the raw production data to train the deep learning model. The input and output can be one or multiple time series. In this work, we investigated the process of downhole pressure prediction using temperature and flow rate data as inputs.

The early stage of deep learning model design is an iterative process. Based on the ideal module characteristic, we can design the main structures for our pressure prediction task. For each model, there are multiple hyperparameters controlling the system. The hyperparameter selection contributes significantly to the model performance, so we need to search and find the best candidates. In the training process, the reservoir properties will be learned and stored in the deep learning structure with tuned hyperparameters. In effect, the deep learning model serves as a simulator. With an arbitrary well data history as an input to the deep learning algorithm, pressure can be estimated according to this given well data history.

Parameters, hyperparameters and deep learning models are three key factors for a successful prediction task. Parameters and deep learning models which need to be determined in the beginning of the model development process, will be discussed in methodologies section; then the hyperparameter tuning process will be discussed in results and analysis section.

Time Series Data Analysis

Data Quality. Two often occurring errors in time series data records are discontinuous measurements and inaccurate measurements. Discontinuous measurement is usually present in the form of time gaps in the well data history, which may be caused by instrument shut down and/or human errors. To avoid the effect of time gaps, it is very important to include the absolute time or time interval as input parameters. In addition, inaccurate measurement may be due to pressure responses that do not follow flow rate behavior, perhaps due to a lack of synchronization between the gauges, or by the absence of flow rate measurements when well condition changes occurred. Though the deep learning model itself has certain tolerance to moderate errors, it is still crucial to train the deep learning model with a good dataset.

Time Sampling Frequency. In many areas, people often use a constant sampling frequency. However, pressure history data may be a combination of short-term transients due to well closure and reopening and long-term transients due to reservoir decline. Using constant time steps of insufficient frequency, rapid pressure variation may not be captured precisely (Figure 1). Ultimately, the pressure points corresponding to shut-in operations will be regarded as outliers by the neural network. If we use extra fine time steps such as 0.02 hour for the entire dataset, the prediction can become unstable and may take longer during training (as observed in this study).

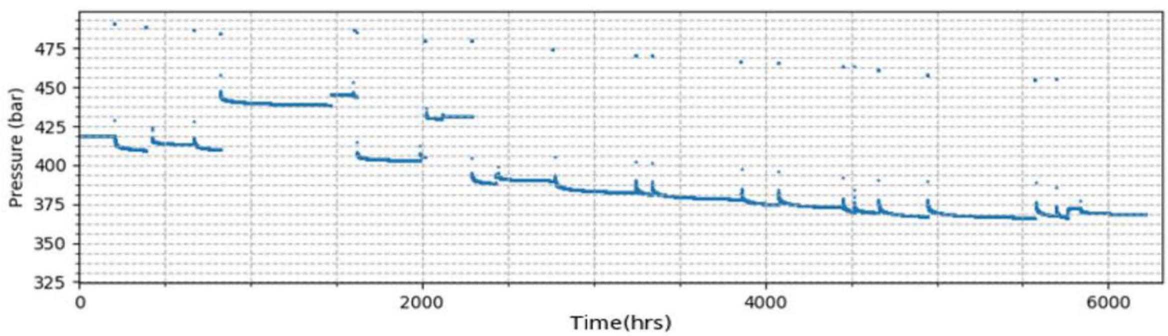

Figure 1—Equal Time Steps

An alternative way is to sample the time series data unequally based on the data characteristics. Figure 2 shows a better way to keep the information from both the transient data and daily to monthly production data, as there are more data sampled in the fast transient period. Besides, as mentioned previously, we can use the time interval as an input parameter to deal with unequal time steps or time gaps.

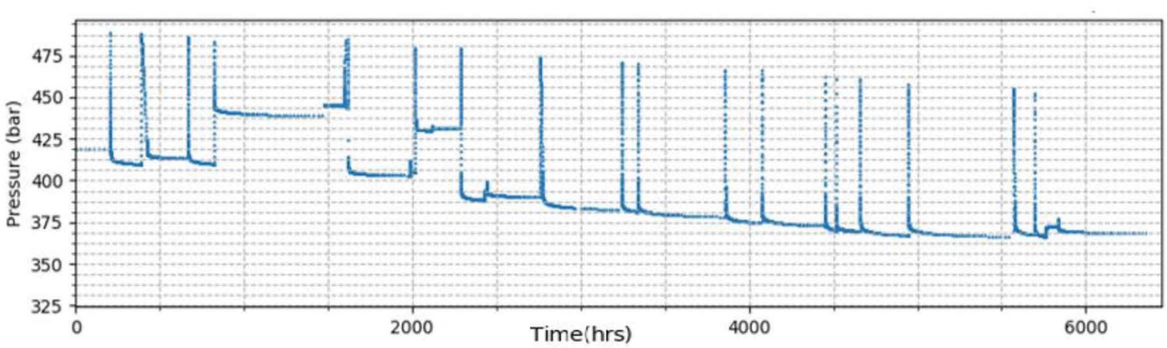

Figure 2—Unequal Time Steps

Input Data format. Example time series parameters that are often measured in the oilfield are listed in Table 1. Here we categorized them into three main groups: pressure, temperature and rate. Temperature data are easy to be acquired. Temperature data have been measured by downhole or surface gauges since their initial installation. However, the temperature data have not been taken good advantages of in early research. A comprehensive investigation of the temperature data suggests that temperature variation is a great indicator of pressure change (Tian and Horne, 2015). Instead of using flow rate data only, we can further incorporate temperature to perform well test analysis and production history analysis. With the temperature to pressure model, we can even trace back to the day when only temperature data is available.

Table 1—Example of Field Data

Input	Temperature	AVG_DOWNHOLE_TEMPERATURE
	Rate	BORE_OIL_VOL
		BORE_GAS_VOL
		BORE_WAT_VOL
		AVG_CHOKE_SIZE_P
		ON_STREAM_HRS
Output	Pressure	AVG_DOWNHOLE_PRESSURE

Sometimes oil rate is not measured directly from each individual well but is an allocated value calculated from the flow rate of the header or main pipeline. Thus, instead we can use choke opening percentage as a more accurate representation of the flow rate of each well. Besides, on-stream time and water cut are also associated with pressure variations. Input combinations can be selected from the available parameters.

When we want to investigate the relationship only between flow rate and pressure, we should only use flow rate as input to the deep learning model. Then, the deep learning model can serve as a simulator to estimate pressure from any given flow rate input. Thus, we can calculate pressure response from a constant flow rate thereby achieving the process of deconvolution. In Table 2, we show six different features associated with flow rate. The first two features, flow rate (q) and flow rate change (Δq), are the traditional features for time series analysis. Then because we are using unequal time steps, we should also include time difference (Δt). One issue to be mentioned is that sometimes we can also add absolute time (t) to capture the overall trend of pressure depletion. To work with a convolution layer in our neural networks, we also utilized features originating from feature-based machine learning as shown by Tian and Horne (2015), which synthesizes physical models and deep learning techniques. In this research, we have also investigated the relationship of the number of features used and the improvement of the prediction accuracy.

Table 2—Flowrate Data Features

# Features	1	2	3	6
Input	q	q	q	q
		Δq	Δq	Δq
			Δt	Δt
				Δq*Δt
				Δq/Δt
				Δq*log(Δt)
Output	pressure			

Output Data format. In general, there are two ways to predict pressure. The first one is to predict pressure change (Δp) and then convert it back to absolute pressure (p). From experiments, we observed that the prediction tightly matches the ground truth. However, a slight deviation in pressure difference prediction will provoke prediction failure as time accumulates. Therefore, we predicted pressure (p) directly in our Deep Learning models.

Deep Learning Models

Deep learning (DL) is a subdivision of machine learning (ML). To achieve more accurate abstractions of data, DL employed the deep neural network structures with a collection of algorithms, which contain various nonlinear transformations. Recurrent Neural Network (RNN) is a prominent invention of deep neural networks which are designed for the sequential characteristics of inputs. Such inputs can be anything where the occurrence of a component in the input series rely on the components ahead of it. (Brandenburg, 2017) For example, speech record, document text and data measured from oilfield sensors can all be considered as sequential data.

Another branch of deep neural networks is the Convolutional Neural Network (CNN), which is often utilized for image processing. CNN models have shown excellent performance for object detection by feature extractions from proper combination of convolution layers and pooling layers. In this work, we also explored an innovative way to combine RNN and CNN for time series prediction.

LSTM & GRU. Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), have been shown to be able to achieve successful results in many applications with time series or sequential data, including weather forecasting, stock market analysis, speech to text generation and other times series data measured from sensors.

RNNs (Figure 3) have several powerful and convenient properties. The length of input sequence is not limited and the RNN model dimension does not expand or shrink for an arbitrary sequence length. (Mohammadi, et al. 2019) More importantly, they can achieve high-level prediction preciseness by capturing long-term temporal dependencies.

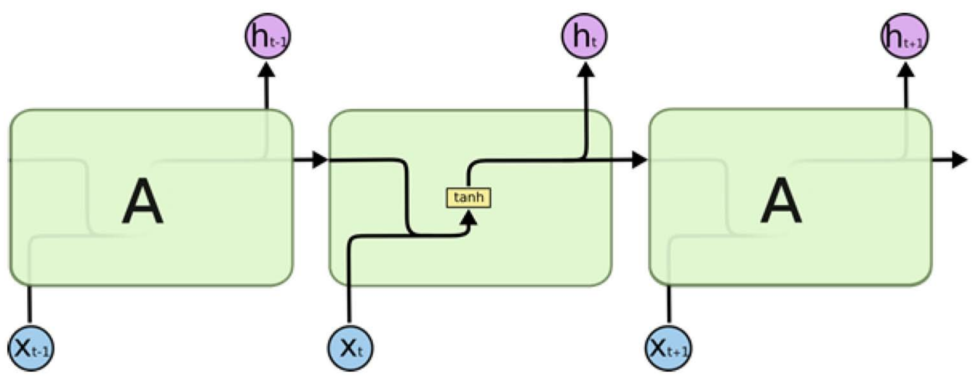

Figure 3—Basic Recurrent Neural Network (RNN) Structure (Olah, 2015)

However, though RNNs are relatively robust, they still confront the obstacles from vanishing gradient, which impedes them from exploiting long term information. Simple RNNs are suitable for storing moderate amount of instances from prior iterations in memory but not for enormous number of instances. Therefore, we employed sophisticated variations of RNNs: LSTM and GRU.

Here, we discuss the design and intuition of a complex activation unit adapted from the basic RNN architecture. LSTMs (Figure 4) are created with persistent memory which enables RNN to obtain and store long-term dependencies.

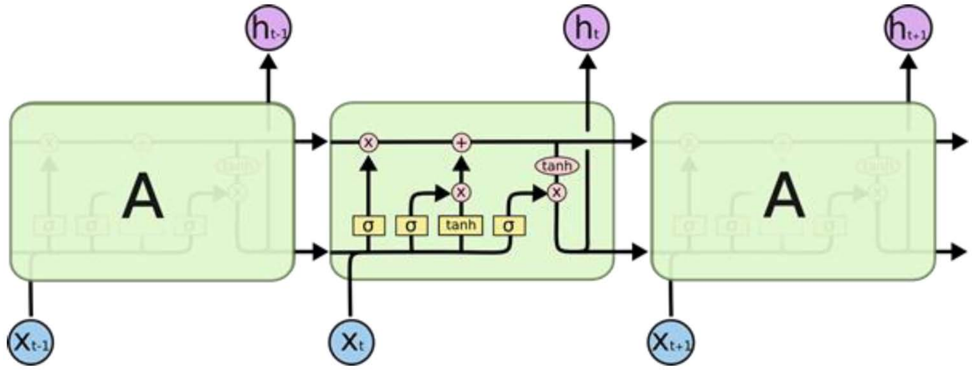

Figure 4—Long-Short Term Memory (LSTM) (Olah, 2015)

First, we have the forget gate. This gate determines whether information should be discarded or retained. Information from both prior new input $x^{(t)}$ and hidden state h^{t-1} is processed by the sigmoid activation function (Eqn. 1). The information will be abandoned if the values $f^{(t)}$ is closer to 0; conversely information will be stored if $f^{(t)}$ is closer to 1.

$$f^{(t)} = \sigma\left(W_f h^{(t-1)} + U_f x^{(t)} + b_f\right) \tag{1}$$

Then we have the input gate $i^{(t)}$ which is also calculated from a sigmoid function (Eqn. 2) to update the cell state $c^{(t)}$ (Eqn. 3,4). The input gate evaluates the importance of input to be carried to the next cell.

$$i^{(t)} = \sigma\left(W_i h^{(t-1)} + U_i x^{(t)} + b_i\right) \tag{2}$$

$$\tilde{c}^{(t)} = \tanh\left(W_c h^{(t-1)} + U_c x^{(t)} + b_c\right) \tag{3}$$

$$c^{(t)} = f^{(t)} \circ c^{(t-1)} + i^{(t)} \circ \tilde{c}^{(t)} \tag{4}$$

Finally, we have the output gate $o^{(t)}$ (Eqn. 5). The output gate controls output content for the current hidden state $h^{(t)}$ (Eqn. 6).

$$o^{(t)} = \sigma\left(W_o h^{(t-1)} + U_o x^{(t)} + b_o\right) \tag{5}$$

$$h^{(t)} = o^{(t)} \circ tanh \; c^{(t)} \tag{6}$$

These gates make assessments on whether input data in a sequence are valuable to be retained or worthless to be dumped. Thereby, the network can preserve relevant information to the downstream.

The GRU (Figure 5) is a recent development of Recurrent Neural networks. There are a lot of similarities between LSTM and GRU. One main difference is that there are only two gates, reset gate and update gate, in a GRU cell. The function of the update gate in GRU is comparable to the function of forget and input gate of LSTM. The update gate $u^{(t)}$ (Eqn 7) determines whether to pass the new information.

$$u^{(t)} = \sigma\left(W_u h^{(t-1)} + U_u x^{(t)} + b_u\right) \tag{7}$$

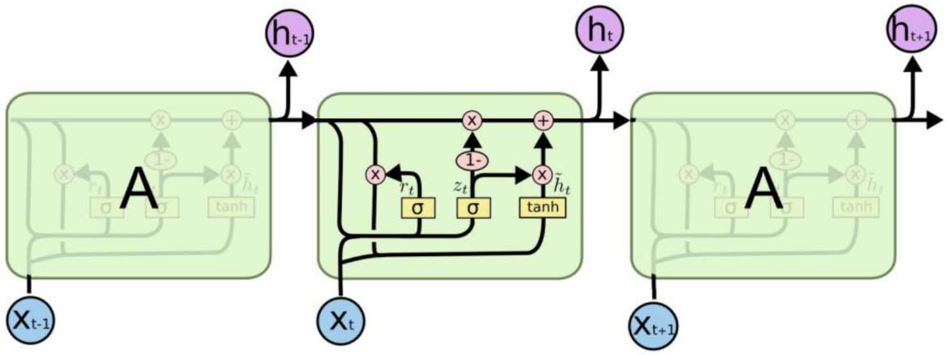

Figure 5—Gate Recurrent Unit (GRU) (Olah, 2015)

The reset gate $r^{(t)}$ (Eqn 8) is then employed to determine to what extend should we forget the past information.

$$r(t) = \sigma(W_r h^{(t-1)} + U_r x(t) + b_r \tag{8}$$

Besides, GRUs used hidden state (Eqn 9,10) instead of the cell state to pass information and exposed the complete hidden content without controlling gates.

$$\tilde{h}^{(t)} = \tanh\left(W_c\left(r(t) \circ h^{(t-1)}\right) + U_h x(t) + b_h\right) \tag{9}$$

$$h^{(t)} = \left(1 - u(t)\right) \circ h^{(t-1)} + u(t) \circ \tilde{h}^{(t)} \tag{10}$$

The mathematically more concise framework compared to LSTM enables GRU to be more efficient computationally.

The RNN blocks can be used individually or jointly. We can also stack a LSTM layer on top of a GRU layer, which can be easily achieved in program libraries such as Keras or Pytorch by using a sequential command. Then we can build the full pipeline of our model.

At time step t_i, given input x_i, where x_i can be:

$$x_i = [(\mathrm{q})_i \quad (\Delta\mathrm{q})_i \quad (\Delta\mathrm{t})_i], \quad i = 1, \ldots, n \tag{11}$$

$$x_i = \left[(\mathrm{q})_i \quad (\Delta\mathrm{q})_i \quad (\Delta\mathrm{q}\cdot\Delta\mathrm{t})_i \quad \left(\frac{\Delta\mathrm{q}}{\Delta\mathrm{t}}\right)_i \quad (\Delta\mathrm{q}\cdot\log(\Delta\mathrm{t}))_i\right], \quad i = 1, \ldots, n \tag{12}$$

$$x_i = \left[(\mathrm{Temperature})_i \quad (\mathrm{Qi})_i \quad (\mathrm{Qw})_i \quad (\mathrm{Qg})_i \quad (\mathrm{Choke\ size})_i\right], \quad i = 1, \ldots, n \tag{13}$$

We want to predict:

$$y_i = p_i, \quad i = 1, \ldots, n \tag{14}$$

In Figure 6 is complete deep learning architecture. First data will go through the RNN layer(s). We can use simple RNN, single LSTM, single GRU or combination modules. Then, as we want to predict one output signal (which is pressure), here we added a fully connected or dense layer which maps 512 values down to only one value. We will discuss the model performance in the results section.

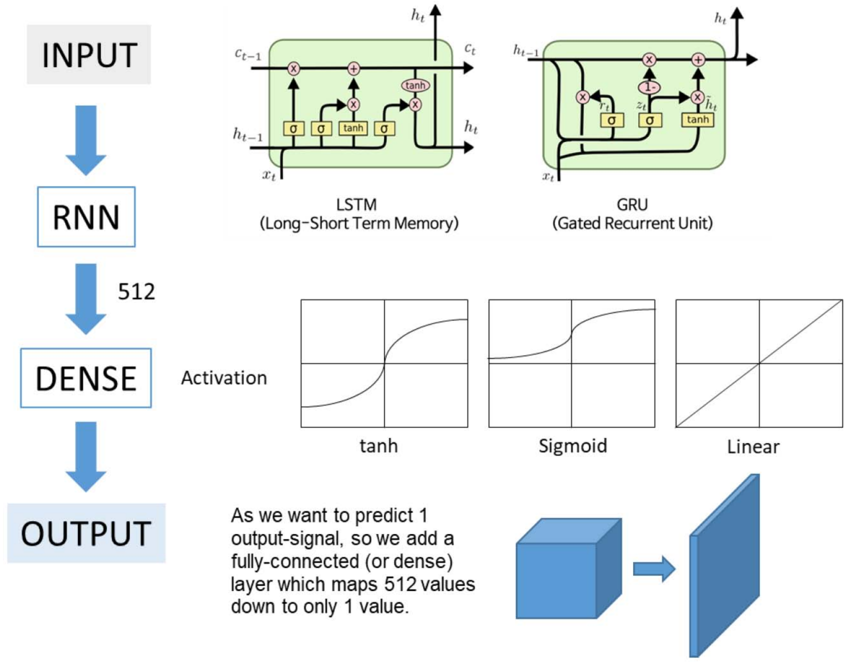

Figure 6—Final structure

LSTNet. There are many innovative combinations of CNN and RNN proposed by researchers recently. In this type of configuration, CNN can be used to identify spatial structures and extract short-term local dependency patterns from the sequences along the feature and time dimensions. RNN helps to recognize long-term patterns from sequential information. Here we utilized the Long- and Short-term Time-series network (LSTNet) structure in Figure 7 (Lai et al., 2018) with adjustments for well data analysis.

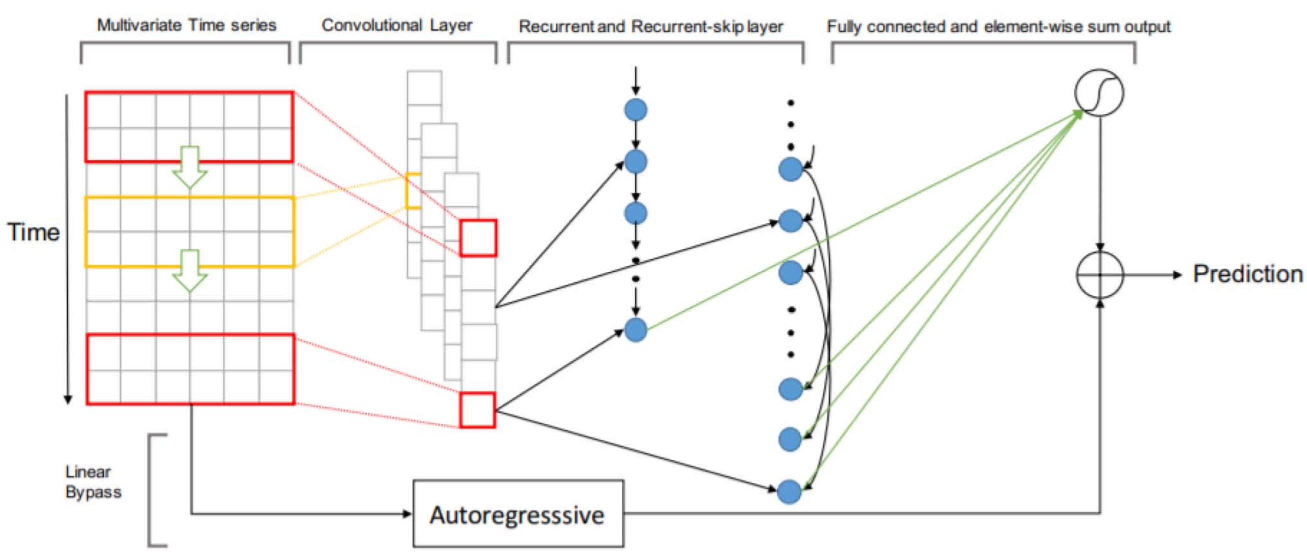

Figure 7—LSTNet Structure (Lai et al., 2018)

Given well signal sequences X (Eqn 15), we aim at predicting the pressure value p_T. Similarly, each xi in X can be the form of xi in Eqn 11, 12, 13.

$$X = \{x_1, x_2, \ldots, x_T\}, \ x_T \in R^n \tag{15}$$

In the first step, the input data matrix is flattened into multiple vectors in the convolution layer. The size of kernel is number of input features times number of past time steps used. To make sure the input and output are the same length, we added padding at the beginning of the input data. We can also perform x_i to p_i prediction by simply setting the vertical kernel window size to one. This is slightly different to the design in the original LSTNet structure. In addition, as we want to capture full information, we used a stride equal to one. Unlike ConvNet, we are not dealing with pictures so there is no pooling layer in this structure. For the k-th filter:

$$h_k = RELU\left(W_k \cdot X + b_k\right) \tag{16}$$

In the second step, data output from convolution layer will be sent to the RNN layer. In the RNN layer, the recurrent-skip structure is designed to capture the periodic pattern. For example, if we have offshore wells which may be affected by tides, we can use the recurrent-skip structure to remove the tidal effect. For normal onshore wells without seasonal patterns, we can use a temporal attention layer or just simply remove the recurrent-skip structure.

Additionally, in LSTNet structure, a traditional linear autoregressive model is incorporated to the nonlinear neural network part simultaneously to improve model sensitivity. Thereby the combination architecture is more powerful for the time series.

Evaluation Indices. Relative squared error (RSE) can reduce model errors to the equivalent dimensions by normalizing the total squared error. Therefore, RSE was utilized to evaluate our models consistently.

$$RSE = \frac{\sum_{i=1}^{n}(pi - ai)^2}{\sum_{i=1}^{n}(\bar{a} - ai)^2} \tag{17}$$

Additionally, we have also calculated relative absolute error (RAE) for reference.

$$RAE = \frac{\sum_{i=1}^{n}|pi - ai|}{\sum_{i=1}^{n}|\bar{a} - ai|} \tag{18}$$

Results and Analysis. The Norwegian company Equinor has released a complete dataset from a North Sea oil field for research, study and development purposes. Volve field was discovered in 1993 and approved for production from 2005 to 2016. (Equinor, 2018) The main production was from 9022 ft to 10236 ft depth of the Hugin Formation sandstone layers, which was formed during the Jurassic age. The dataset contains well logs, drilling and completion data, subsurface seismic data and production data

We utilized the production data of well NO 15/9-F-1 C from April 2014 to April 2016 to test our Deep Learning models. The available well production dataset include the average downhole pressure (Figure 8-a), average downhole temperature (Figure 8-b), oil flow rate (Figure 8-c), gas flow rate (Figure 8-d), water flow rate (Figure 8-e), on-stream hours (Figure 8-f) and choke size percentage (Figure 8-g). Data were split into 70%: 15%: 15% for training, validation and testing throughout different model experiments. RSE and RAE were calculated for model comparison.

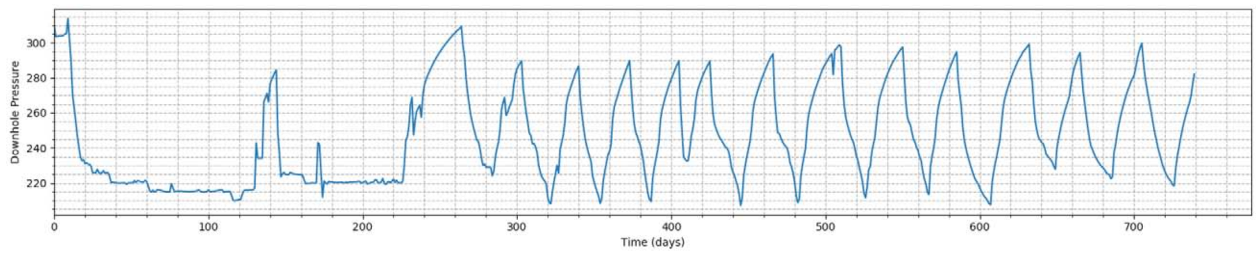

Figure 8-a—Volve Field – Average downhole pressure

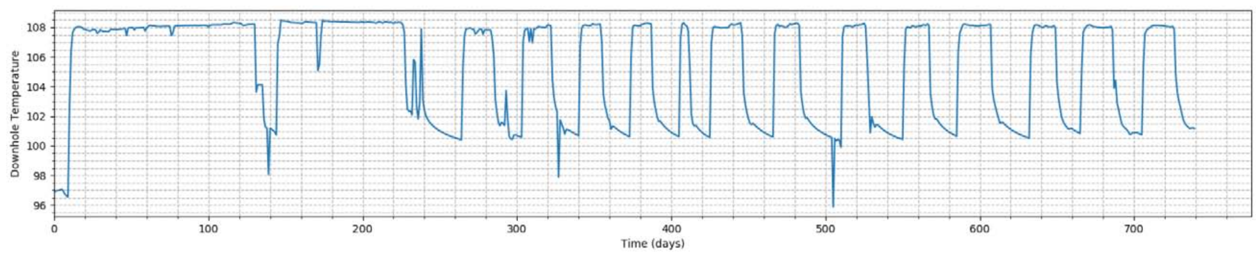

Figure 8-b—Volve Field – Average downhole temperature

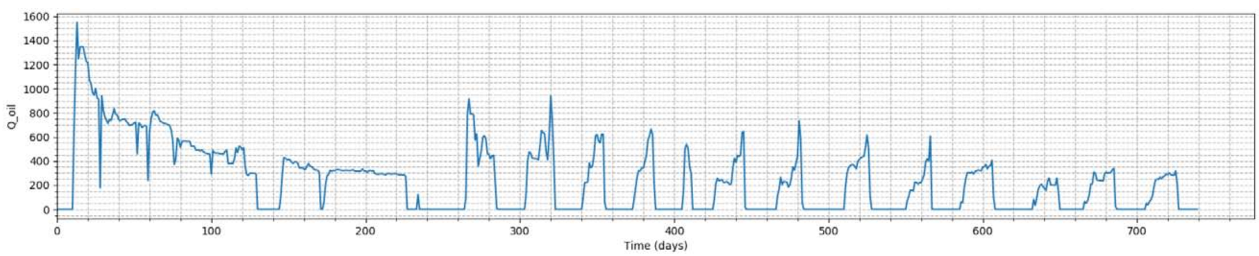

Figure 8-c—Volve Field – Oil flow rate

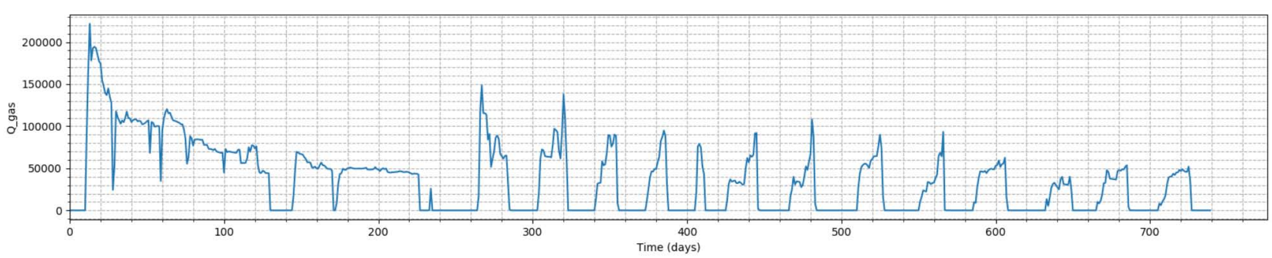

Figure 8-d—Volve Field – Gas flow rate

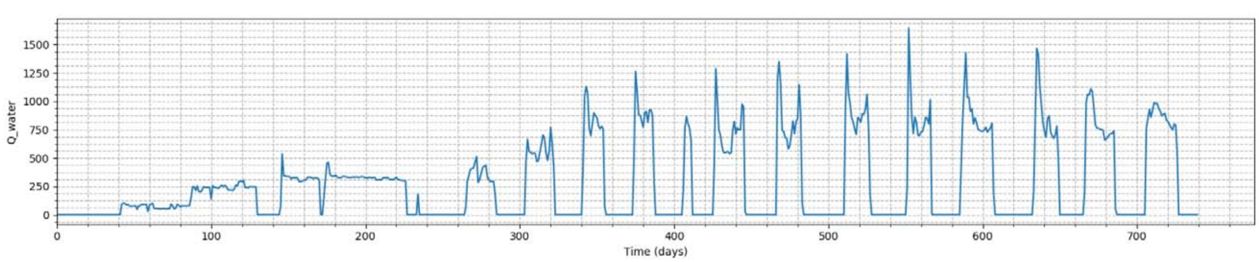

Figure 8-e—Volve Field – Water flow rate

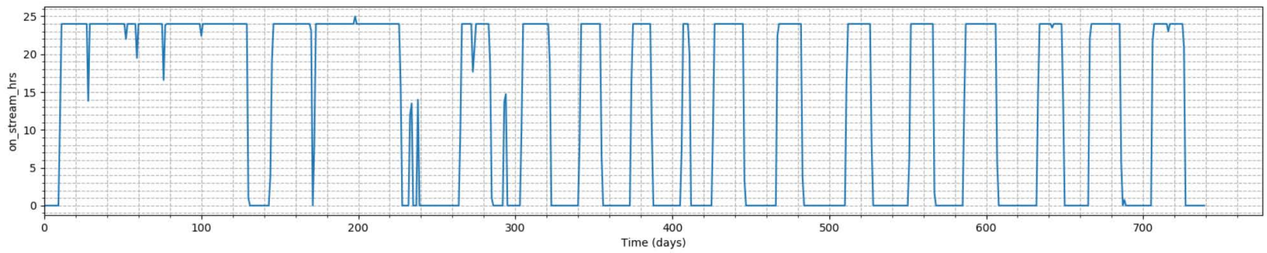

Figure 8-f—Volve Field – On-stream hours

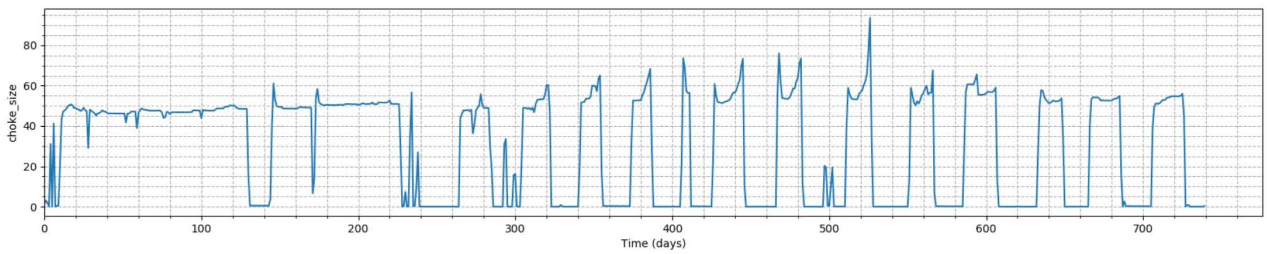

Figure 8-g—Volve Field – Choke size percentage

Pressure Prediction

First we performed the feature analysis based on the structure in Figure 6 to find the best feature combination for pressure estimation. Then we analyzed the different model performance in terms of time and accuracy.

Feature analysis

Table 3 shows the prediction results using single parameters based on a single GRU layer structure. In this structure, the GRU module contains 64 units. The activation function for the dense layer is the linear function. Batch size is eight and sequence length is two weeks.

Table 3—Single features

Input	RSE_test	RAE_test
Flow rate (gas)	0.8803	0.9376
Flow rate (oil)	1.1260	1.0700
Flow rate (water)	0.3116	0.5289
Temperature	**0.1995**	**0.3972**
On stream hours	0.2214	0.4392
Choke opening percentage	0.1962	0.4081

We found that the accuracy is higher by using water flow rate only compared with using gas or oil flow rate. This is because water is the main production in this multiphase production system. Besides, as expected, choke opening percentage is actually a better representation of the overall flow rate. The overall error reduced significantly by using choke instead of direct flow rate record. Moreover, from our observation, on-stream hours is also a good indicator of the daily-based average downhole pressure. Among all others, the best individual candidate for pressure prediction was found to be temperature. Then, we combined best performing parameters to further improve prediction accuracy.

As we discussed in the previous section, we could benefit from the features designed for traditional machine learning models. In Table 4, the q represents three data sequences: oil, water and gas flow rates. The results show that the performance of the model is already good when using three features. This indicates that the deep learning model can correctly extract the relationship between flow rate and pressure in the physical model without using hand-crafted features.

Table 4—Flow rate features

Input						RSE_test	RAE_test
q						0.2067	0.4514
q	Δq					0.1794	0.4031
q	**Δq**	**Δt**				**0.1740**	**0.4115**
q	Δq	Δt	Δq*Δt	Δq/Δt	Δq*log(Δt)	0.1767	0.3213

Then we performed the pressure prediction with feature combinations.

Table 5—Feature combinations

Input	RSE_test	RAE_test
Q_interval_g, Q_gas	0.7072	0.8003
Q_interval_o, Q_oil	0.9493	0.9355
Q_interval_water, Q_water	0.2364	0.4591
Q_gas, Q_oil, Q_water	0.2067	0.4514
time_interval, Q_interval_gas, Q_interval_oil, Q_interval_water, Q_gas, Q_oil, Q_water	0.1740	0.4115
Temp_in, time_interval, Q_interval_gas, Q_interval_oil, Q_interval_water, Q_gas, Q_oil, Q_water	**0.1394**	**0.3631**
Temp_in, time_interval, choke_size Q_interval_gas, Q_interval_oil, Q_interval_water, Q_gas, Q_oil, Q_water	0.3199	0.6222

Model Structure

After finalization of the feature combinations, we can use them to test our models. Splits of train: validation: test = 70:15:15 are shown in black dash lines in Figure 9 and subsequent plots.

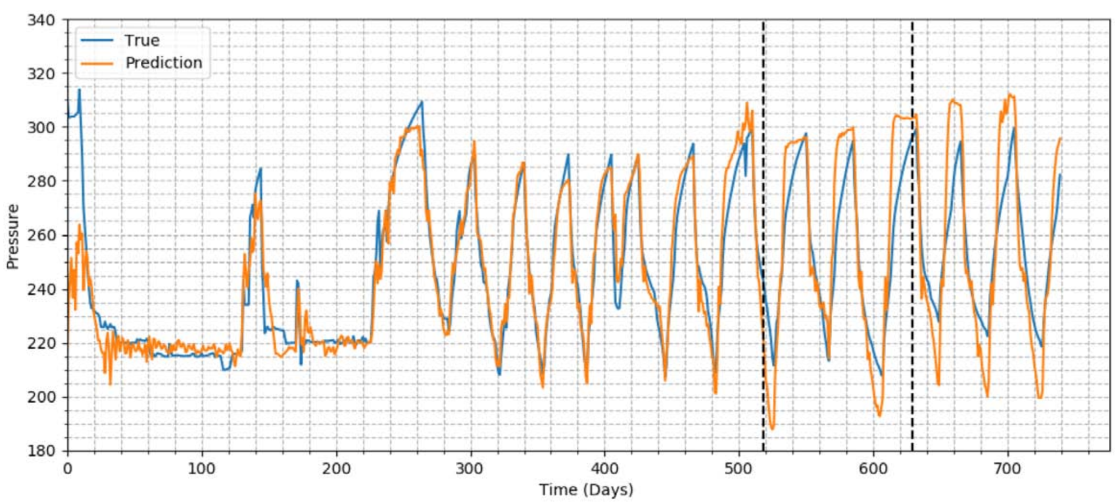

Figure 9—Pressure Prediction by Simple RNN

From Table 6, we find that both LSTM and GRU provided better estimation compared to simple RNN. Also, GRU is better compared to LSTM in terms of speed and accuracy. Besides, comparing Figure 10 and Figure 11, GRU is capable of capturing the early noisy behavior. The combination of LSTM and GRU does not enhance the overall precision but prolongs the training time.

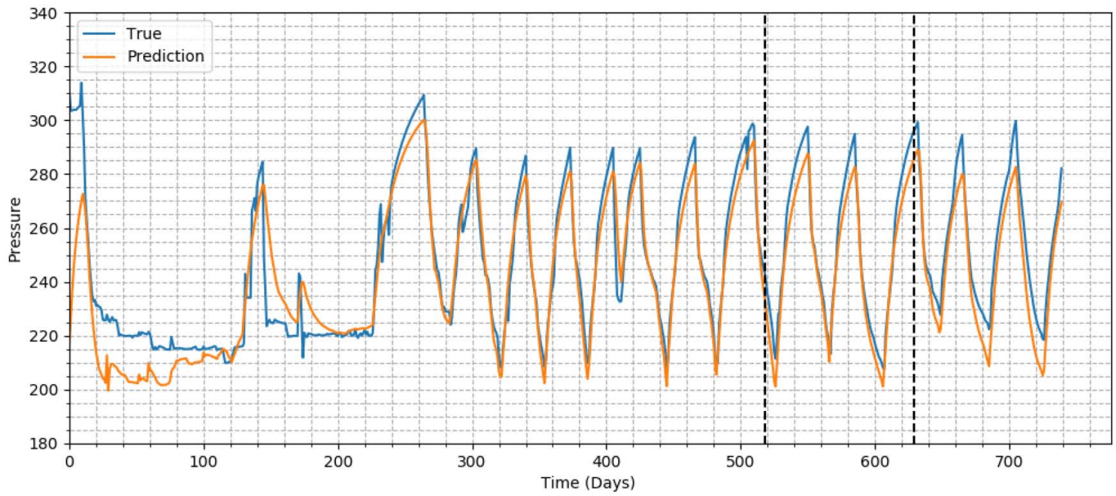

Figure 10—Pressure Prediction LSTM

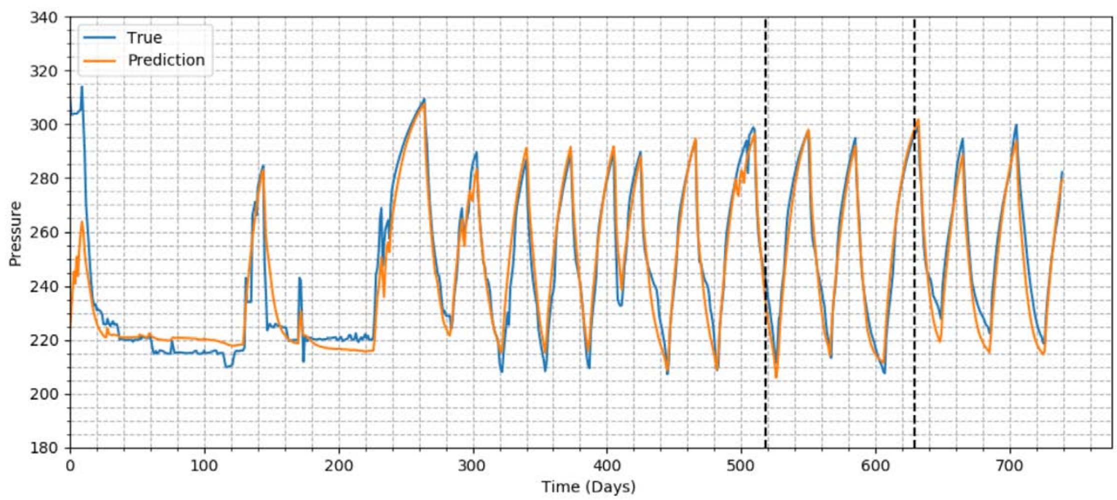

Figure 11—Pressure Prediction GRU

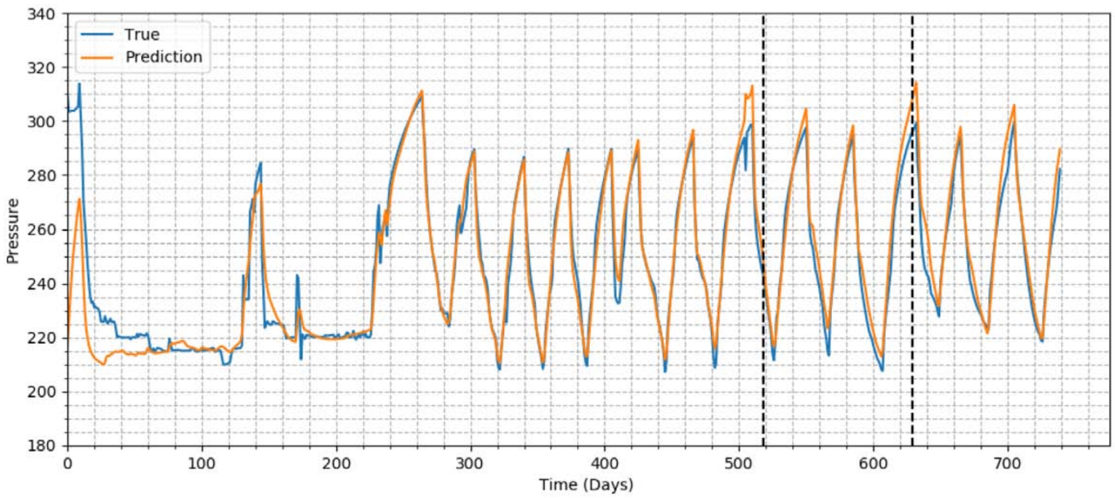

Figure 12—Pressure Prediction LSTM+GRU

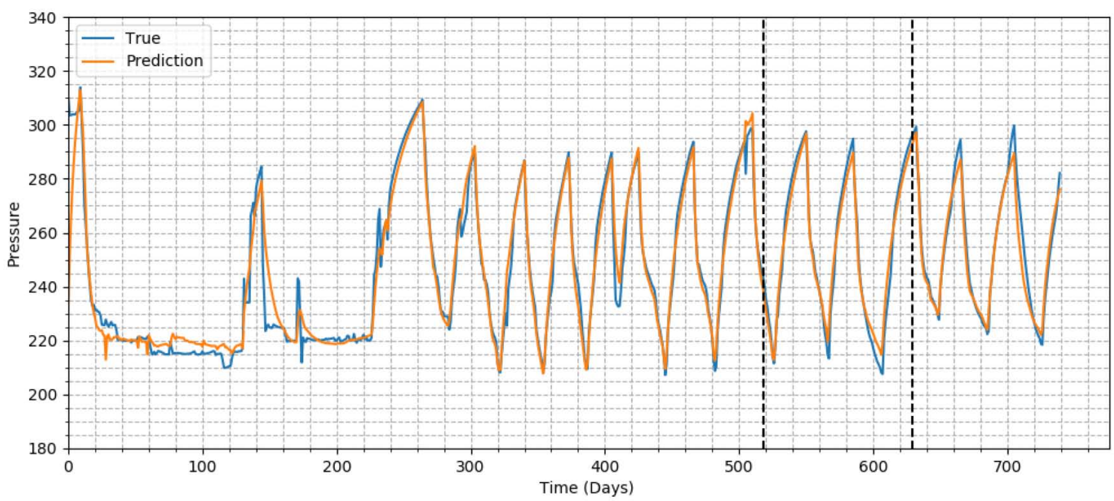

Figure 13—Pressure Prediction LSTNet

Table 6—Model Analysis

Structure	Time for each case (s)	RSE_test	RAE_test
RNN (Simple RNN)	15.14357	0.5739	0.7389
RNN (GRU)	178.0561	0.1394	0.3631
RNN (LSTM)	292.6135	0.2008	0.4584
RNN (GRU + LSTM)	920.3116	0.1145	0.3211
LSTNet (CNN+RNN)	523.4847	0.0884	0.2694

After a long hyperparameter tuning process, the LSTNet was found to the yield best RSE and RAE. Hyperparameters are the adjustable variables controlling the neural network. In LSTNet, one key hyperparameter is the number of past values used for the new prediction. From several experiments we found that the model generates the optimal result with five steps ahead. This is reasonable considering that CNN is designed to capture local behaviors, so the number of past values should not be too long.

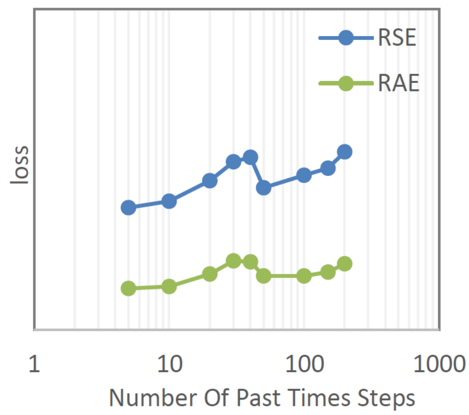

Figure 14—Kernel size tuning

Conclusion

In production data, parameters are usually well aligned at each time step, so we can easily employ deep learning approaches without an arduous hand-labelling process. Each well data history contains two patterns. The short-term pattern portrays the high frequency pressure transient data corresponding to buildups from well shut-ins, while the long-term pattern reflects the daily to monthly production. A successful time series forecasting model should be able to capture both kinds of recurring patterns for accurate predictions. In this work, we investigated two sequential deep learning structures for well data history analysis. Both GRU and LSTM are more accurate than a simple RNN structure. GRU is slightly faster than LSTM as the number of gates is less that in LSTM. For more complicated field cases, LSTNet demonstrated significant performance improvements over the baseline methods. We explored pressure prediction only, however this work can also be extended to the prediction of flow rate or temperature.

Acknowledgement

We would like to thank the Smart Fields Consortium for the financial support that made this research possible.

References

An, P., & Moon, W. M. (1993, January 1). Reservoir Characterization Using Feedforward Neural Networks. *Society of Exploration Geophysicists*.

Brandenburg, Justin. "Applying Deep Learning to Time Series Forecasting with TensorFlow." *MapR*. June 10, 2017. https://mapr.com/blog/deep-learning-tensorflow/.

Essenreiter, R., & Karrenbach, M. (1996, January 1). Deconvolution Using Neural Networks. *Society of Exploration Geophysicists*.

Equinor. "Volve Data Village." *Equinor Data Portal Beta*. October 18, 2018. https://data.equinor.com/dataset/Volve.

Hwang, Tim. *"Computational power and the social impact of artificial intelligence".* arXiv preprint arXiv:**1803**.08971 (2018).

Lai, Guokun, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. "Modeling long-and short-term temporal patterns with deep neural networks." In The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, pp. 95–104. ACM, 2018.

Li, Yuanjun, Andrei Popa, Andrew Johnson, Iraj Ershaghi, and Steve Cassidy. "Dynamic Layered Pressure Map Generation in a Mature Waterflooding Reservoir Using Artificial Intelligence Approach." In SPE Western Regional Meeting. Society of Petroleum Engineers, 2018.

Mohammadi, Milad, Rohit Mundra, Richard Socher, Lisa Wang, and Amita Kamath. *"CS224n: Natural Language Processing ... - Web.stanford.edu".* January 2019. http://web.stanford.edu/class/cs224n/readings/cs224n-2019-notes05-LM_RNN.pdf.

Olah, Christopher. "Understanding LSTM Networks." *Understanding LSTM Networks -- Colah's Blog*. August 27, 2015. http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Tian, Chuan, and Roland Horne. "Applying Machine-Learning Techniques to Interpret Flow-Rate, Pressure, and Temperature Data from Permanent Downhole Gauges." *SPE Reservoir Evaluation & Engineering* (2019).

Tian, Chuan. *Machine Learning Approaches for Permanent Downhole Gauge Data Interpretation*. PhD Diss. Stanford University, 2018.