International Journal on Advanced Science Engineering Information Technology

Vol.13 (2023) No. 4
ISSN: 2088-5334

Performance Analysis of Deep Learning Implementation in Operational Condition Forecasting of a Gas Transmission Pipeline Network

Aditya Firman Ihsan [a,*], Darmadi [b], Saladin Uttunggadewa [c], Silvya Dewi Rahmawati [d], Irsyad Giovanni [b], Salamet Nur Himawan [b]

[a] *School of Computing, Telkom University, Bandung, Indonesia*
[b] *RC-OPPINET, Institut Teknologi Bandung, Bandung, Indonesia*
[c] *Mathematics Department, Institut Teknologi Bandung, Bandung, Indonesia*
[d] *Petroleum Engineering Department, Institut Teknologi Bandung, Bandung, Indonesia*
Corresponding author: [*] adityaihsan@telkomuniversity.ac.id

***Abstract*— Monitoring natural gas transmission in a pipeline network is important to maintain the supply and demand balance in natural gas transactions and distribution. Gas pressure, temperature, flowrate, and gas properties must be monitored during the transmission process. These variables, also known as operational conditions, need to be simulated carefully to understand the dynamics and behavior over time. Commonly used physical equations, such as thermodynamic or hydraulic equations, have limitations in simulating future trends because they need some known boundary conditions to be solved. In that case, data-driven method is needed, especially nowadays when data management is widely implemented. This paper implements a deep Recurrent Neural Network (RNN) to forecast the future behavior of gas pressure as an operational condition in a gas pipeline network receiving platform. Different types of recurrent cells are used, i.e., Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU). The model is trained in 8 years of a gas pipeline network operational data. Historical flowrate data in the end-nodes become the forecast input in addition to the past pressure data. The sensitivity of the model and learning parameters is experimented with and analyzed to understand the capacity of the RNN in the given task. Mean absolute error is set as the satisficing metric, whereas the training time is set as optimizing metrics. The obtained best model successfully forecasts the future pressure of one day ahead with only around 2% relative error.**

***Keywords*— Recurrent neural network; operational condition; forecasting; gas pipeline network.**

Manuscript received 21 Aug. 2022; revised 6 Apr. 2023; accepted 19 May 2023. Date of publication 31 Aug. 2023.

I. INTRODUCTION

The oil and natural gas industry has many complexities and challenges. Managing the production and distribution process of crude oil or natural gas is important in the increasing need for oil and gas in the energy supply. Specifically, natural gas has been used in various sectors, from households to factories. One of the main challenges in natural gas production is the far, or even sometimes isolated, location of gas sources or wells. For this matter, the gas should have flowed through a long network of pipelines. Monitoring natural gas production in terms of its transmission from suppliers to traders or shippers should be done properly. Gas from production platforms will be pressurized or compressed to flow through the pipeline as it reaches the receiving facility's low-pressure area. Coordinated pressure management between platforms is thus important in ensuring supply and demand balance.

Ideally, a simulation of the flow throughout the pipes is needed for monitoring. However, a lot of coupled variables are involved in the system. By physical law, this simulation is modeled using hydraulic and thermodynamic equations. The system Hydraulic equation describes correlations between the pressure and flowrate of the gas [1]. The coupling of the thermodynamics equations forms a non-isothermal system that involves additional variables, i.e., gas temperature. Solving this system of equations is not an easy task to be done analytically. Thus, numerical approaches are often taken for practicality [2], [3].

Despite its capability for simulating flow, hydraulic equations only govern the flow of one pipeline. In the case of branching or a network of pipelines, additional correlations

and equations are needed, which needs a decoupling trick to be applied [4]. Nevertheless, these physical governing equations still lack the capability for future behavior prediction because the boundary condition of the system at every time step is needed to simulate the whole system completely.

In recent years, the development of data science techniques has presented many data-driven methodologies to be applied to various problems in many sectors. This applicability of data-driven techniques is also true for the oil and gas industry, as some studies have been done in that area, such as predictive maintenance, process optimization, time-series forecasting, and the calculation of fluid properties [5], [6].

As physical systems such as hydraulic and thermodynamics equations have some limitations, some data-driven methods are emerging in the face of the advancement of machine learning, from traditional to contemporary. For example, forecasting of gas shale or wells production can be done using classical statistical methods such as Support Vector Machine (SVM), linear regression, or autoregressive (AR) [7] or the contemporary method such as neural network [8]. Even though some more advanced techniques of deep learning have developed, traditional machine learning techniques such as a nearest neighbor or random forest are still largely used, such as for beam pump dynamometer data classification [9], prediction of dead oil viscosity correlation [10], offshore structure predictive modeling [11], oilfield process control [12], drifting behavior detection [13], and crude oil contamination detection [14].

In the case of neural networks, their advancing techniques, with some modifications and adaptations nowadays, are also widely implemented in many areas of oil and gas. To name a few, some of them are artificial lift selection [15] and correlation development to predict oil rate [16], computation of gas hydraulic [17], failures prediction [18], risk assessment [19], flaring event prediction [20], Enhanced Oil Recovery (EOR) and carbon dioxide storage capacity [21], burst pressure prediction [22], optimization in Pressure Swing Adsorption (PSA) process. Even the advanced generative model, such as GAN (generative adversarial network) can be applied to map water saturation from reservoir properties [23].

For monitoring purposes, the need to be knowledgeable of possible future behavior arises due to the popularity of forecasting tasks as one of the machine learning capabilities. Some aspects that can be forecasted are supply and demand for transmission control [24], flowrate in gas production [25] and oil production [26], crude oil price crash [27], or demand fluctuation for bullwhip assessment [28].

In this study, we utilize a neural network, specifically a recurrent neural network (RNN), to forecast pressure in a gas pipeline network sink node that receives gas flow from 4 different sources. To our knowledge, implementing a neural network for gas pressure monitoring purposes is quite new in the literature, especially the performance analysis to understand the capability of the forecasting model with some given adjustments. Historical flowrate data from all network nodes over time become the input feature and the past pressure of the received gas in the sink node. We set up some cases to see how different neural network architecture affects the forecasting results.

II. Material and Method

A. Dataset

The dataset used in this paper is a pipeline network operational data consisting of physical conditions of the gas, such as pressure, temperature, energy rate, and volume rate, with additional component fraction features of the gas. Because the temperature is relatively stable and almost homogeneous, isothermal system is considered. The fractional data of the gas components is excluded in this case in the assumption of no significant correlation with the dynamics of the gas pressure. Thus, we use only the pressure and flowrate feature of the gas, where the volume rate is used to represent flowrate rather than the energy rate. The data is taken hourly from each network platform (nodes), which are four sources (transmitting facility/reservoir) and 1 sink (receiving facility), in 8 years' time range (2013-2021). In total, that counts around 69576 data points.

In some field cases, pressure in the source nodes is control variable that should be adjusted. Thus, we only use flowrate data from those nodes, resulting in only six features (4 source flowrate, sink flowrate, and sink pressure). Our forecasting target is the sink pressure.

Field dataset usually contains many broken entries or anomalies from many practical errors or failures. These anomalies are not always in the form of empty values but also in the form of unnaturally very high or very low values. However, some caution should be taken in handling the broken datapoints, as it may represent the true condition in the field. For instance, a zero flowrate value in some consecutive datapoints may be caused by a sudden short shutdown from one of the compressors. This kind of data should be included as it still affects the received gas pressure in the Sink. Also, blindly removing the datapoint may cause discontinuity in the series. For that matter, we handle these issues by first carefully detecting outer outliers. Due to the distinct overall behavior of flowrate and pressure, we use different techniques in detecting anomalies: quantile-based selection for the source node data and variance-based selection for the sink node data. All detected outliers are then replaced with the last good value of the data (previous datapoints) after being checked thoughtfully to ensure they are truly anomalies.

After all, anomalies are handled successfully; the time series are smoothed to reduce extreme fluctuations that occur frequently in the data, as shown forming many spikes in Figure 1. In the data profile, we can see that the data is stationary without periodic trends. We apply a forward rolling average with width of 6 hours intervals to smoothen the spikes. This means that the data values in 6 hours is averaged to determine the new value of the 7th hours data. Next preprocessing taken is scaling transformation. To avoid negative values, instead of standard normalization (gaussian transformation), max-min rescaling is applied to squeeze the data to intervals of zero and one.

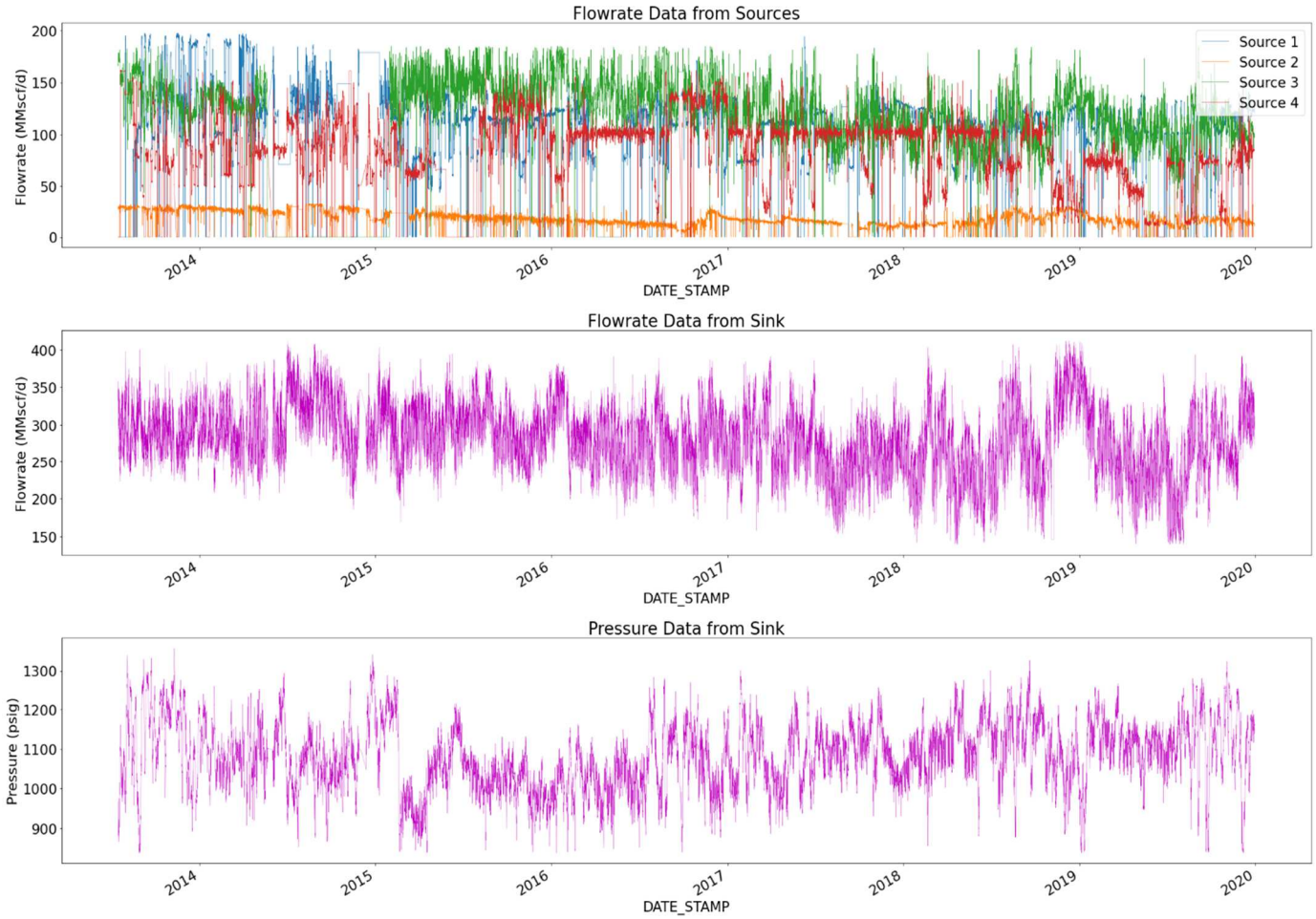

Fig. 1 Profile overview of the dataset

In these three steps, the data is ready to be learned. To finalize the preparation, we split the data into two sets: a training set and a validation set. Standard splitting usually uses randomized sampling with an 8:2 ratio of both sets. In the case of time series data, this splitting is implemented in 'window' level, not datapoint level. It means that the data is cut in consecutive time windows of fixed length to preserve the sequential information of the series. However, in this case, because we still want to track the timestamp of the data, we split the data first based on the timestamp. The last 18 months of the data (Jan '20- July '21) is taken as the validation set and the rest become training set.

After the splitting, the windowing process occurs, in which each data set is bundled in 72 datapoints sequences. To avoid overfitting of the model in learning some regularities, the window is taken every three datapoints interval. In each window, the first 48 datapoints are set to be the forecasting input, and the last 24 datapoints are the output. In other words, we prepare the data to be learned so that one day (24 hours) operational behavior in the future can be determined by the behavior of two days prior (48 hours). After the windowing process, we close the preprocessing by shuffling all the windows and bundling it in batches of 8 windows to be learned by the model.

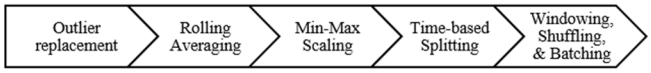

Fig. 2 Data preprocessing steps

B. *Model Architecture*

This study's core model is Artificial Neural Network (ANN). ANN in brief, is a method inspired by biological brains to approximate a function in the form of a set of linear computations. These linear computations, each called neurons, are 'activated' by a nonlinear function to accommodate nonlinearity. The commonly used activation function in ANN is so-called Rectified Linear Units (ReLU) [29].

Neurons can be stacked consecutively to increase the depth of complexity approximated by the ANN. Each neuron has internal linear parameters that are refined iteratively using gradient descent optimization. By computing the approximation loss compared to the expected or true values, the gradient of the loss can also be computed as a guide to adjust the parameter in a way that the loss decreases.

In this study, we use time series data, which is one form of sequential data. One ANN modification that can accommodate sequential data is Recurrent Neural Network (RNN). In RNN, the neuron is modified to a cell that has additional output called memory, which is a part of information of current datapoints that become additional input for the next datapoint. There are many variations of recurrent cells. Some of those are Long Short-Term Memory (LSTM)

cell and Gated Recurrent Units (GRU) cell. LSTM cell accommodates two memory states, i.e., long-term and short-term, with 3 internal computations called gates, i.e., forget gate, input gate, and output gate. Each gate has its functionalities to preserve information passed through from the previous cell to the next cell. GRU cell is a modification of LSTM where the input and forget gates are combined to update the gate. Detailed explanations and mathematical computations inside each of these cells [30].

In this paper, we formulate the architecture in two blocks. The first block, the recurrent block, will learn the data's time-based features, and the second block, the dense block, will process the features to predict future time-step. We use four different recurrent architectures for the recurrent block, i.e., vanilla/plain RNN, LSTM, GRU, and bidirectional LSTM (bi-LSTM). The last architecture is an RNN modification where two recurrent layers are stacked to learn opposite time directions simultaneously. Fully connected linear layers are used with ReLU activation for the dense block. The illustration of the architecture is shown in Fig. 3.

Finally, the model is evaluated using mean absolute error (MAE) as the standard time series data metric. MAE is used during training as a loss function to guide the backward propagation in updating model parameters. We also use relative error (RE), also known as approximation error or percentage error, to obtain errors in original scales relative to the expected value. This error gives more understanding in practical use cases because it illustrates measurement or prediction precision. The relative error is computed as the percentage of the average ratio of absolute difference and the real data. The mathematical formula for MAE and RE is written in (1) and (2).

$$MAE = \frac{1}{n}\sum_i |y_i - \hat{y}_i| \quad (1)$$

$$RE = \frac{1}{n}\sum_i \frac{1}{y_i} \cdot |y_i - \hat{y}_i| \cdot 100\,\% \quad (2)$$

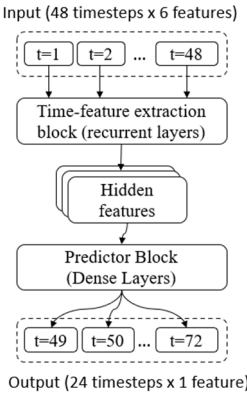

Fig. 3. Model architecture illustration

III. Result and Discussion

A. Case Study

The model built following the general architecture shown in Fig.3 is trained using different case studies. We vary some aspects of architecture and also the learning process. In total, we tried six different experiments, each with 3 or 4 cases. The details of the cases are shown in Table 1. For each variable experiment, other variables are set in the default value: LSTM cell-type, 3 recurrent layers, four dense layers, 512 dense neurons, 100 recurrent cells, and 0.001 for the learning rate.

We aim to observe the sensitivity of the model performance in different setups. We train all model cases in 40 epochs with a learning rate to exponentially decrease as the training progresses. The learning rate in Table 1 is the initial rate before decaying in the second epoch.

TABLE I
Experimental cases

Variable	Case 1	Case 2	Case 3	Case 4
cell-type	LSTM	GRU	BiLSTM	Simple
#recurrent-layers (l)	1	2	3	-
#dense-layers (k)	1	2	3	4
#recurrent-cells (c)	50	100	200	-
#dense-neurons (n)	64	128	256	512
learning-rate (lr)	.005	.001	.0005	.0001

B. Training Performance

During training, we feed the model with validation data at each epoch and compute the MAE to obtain the unbiased state of the model as it progresses. For each experimental case, we plot the profile of MAE calculated in the validation data over training epochs. We compare the profile of all cases for each experiment variable in Fig. 4.

In overview, validation MAE is unstable as it always fluctuates due to uncertainty induced by validation data models has never been seen before. This is not a problem as we seek to see the general trend of the profile. For the first figure, Fig. 4(a), we see that even though the bi-LSTM model has a short plateau during a few first epochs, all the models reached a similar state at the 40th epoch. It may be difficult to see, but the single LSTM model has the most stable convergence in the training process. One possible explanation is that LSTM has more complexity than GRU and Simple RNN. Also, bi-LSTM probably is unsuitable for this kind of time series data because the information is always moving forward without the necessity of future impact on the past. Thus, creating two-directional recurrent layers adds unnecessary complexity to the model, causing it to learn slower.

A common understanding of neural networks is that the model's depth and width increase the model's capacity to learn more abstract information from the data. The number of layers represents the depth of the neural network, and the width is represented by the number of neurons or cells in each layer. However, as presented in Fig. 4(b)-(e), which shows the profile results of the experimental cases of depth and width variation of the model, the validation MAE is almost similar without any significant difference between all cases. The increasing complexity of the neural network does not give better results. In the case of the model width, careful observation shows that the model with 512 neurons or 200 cells, which is the highest value of each experiment, has a more stable performance. Other than that, meaningful insight is difficult to draw.

For the learning rate, as shown in Fig. 4(f), it can be easily seen that the optimal value is 0.001, which gives stable and fast convergence of validation MAE. The lower values, i.e., 0.0005 and 0.0001, result in slower convergence, whereas the higher values, i.e., 0.005, give a more unstable performance. It even fails to reach the expected minimum loss, as the loss suddenly rises around 5th epoch.

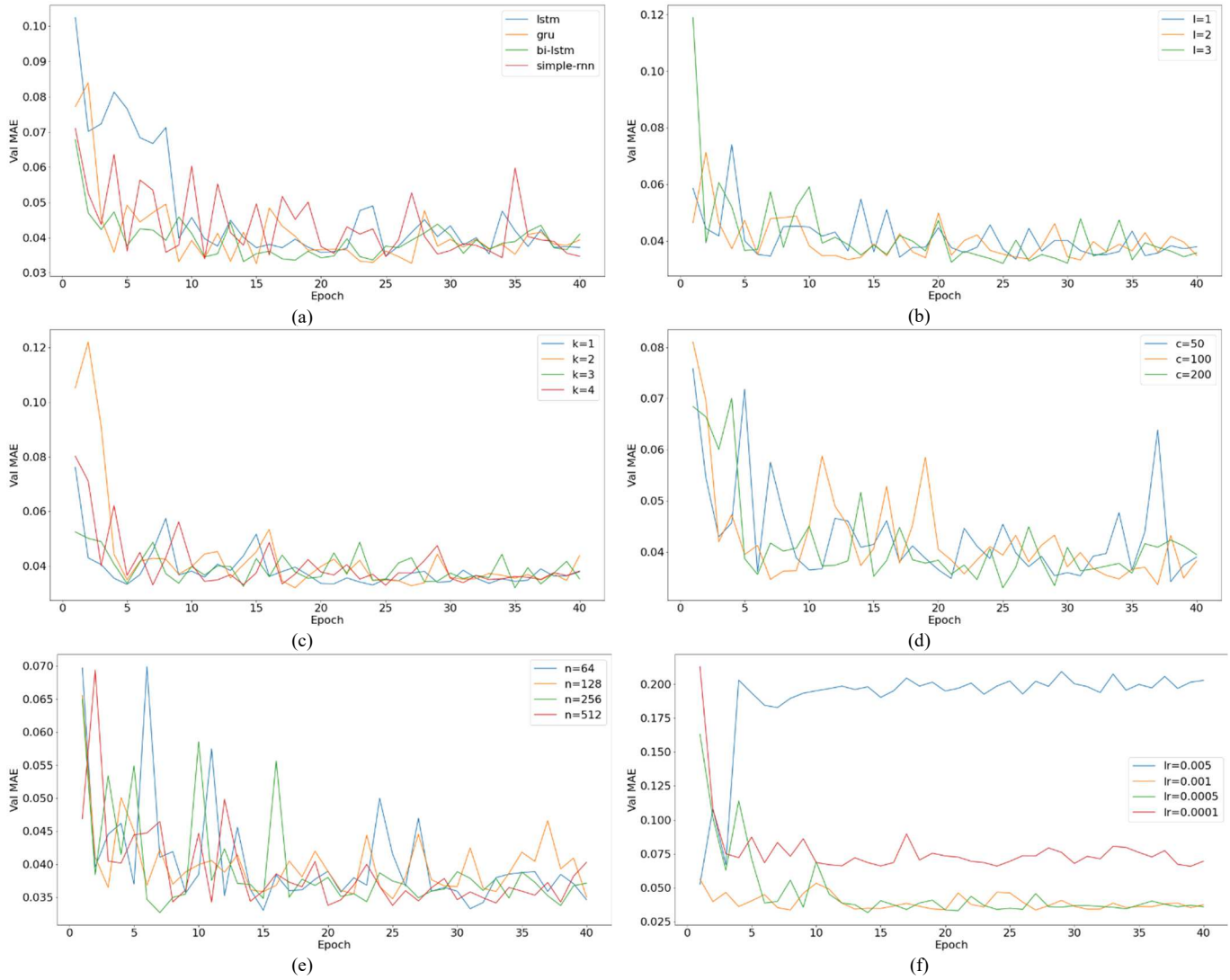

Fig. 4 Validation MAE Profile in all experimental cases

C. Computational Load

In choosing the best model, many aspects have to be considered. Obtaining the smallest possible validation MAE is the main objective. However, other constraints should be considered, such as the load of the model. A heavier model without a significant gain in the result is unnecessary. We need to obtain a 'good enough' model with the proper load. The second metric we use to analyze the model is the training time for every 40 training epochs in each case. The results are shown in Table II.

TABLE II
TRAINING TIME OF ALL CASES

Variable	The training time of 40 epochs (second)			
	Case 1	Case 2	Case 3	Case 4
cell-type	758.9	**683.7**	1612.8	1761.7
#recurrent-layers (l)	**696.0**	765.5	817.45	-
#dense-layers (k)	851.9	**812.7**	814.4	817.5
#recurrent-Cells (c)	707.7	710.5	**699.9**	-
#dense-neurons (n)	745.9	768.5	673.1	**659.4**
learning-rate (lr)	921.8	818.3	**817.5**	819.1

As seen in the previous result, the validation MAE ends up indifferent for all cases, where eventually, a value around 0.37 is reached. In that case, we swap the role by setting validation MAE as the satisficing metric and the training time as the optimizing metric. The main objective of choosing the best model is obtaining the shortest training time possible.

In Table II, we see the training time in the second of each case. The gap between cases is quite high for the cell-type experiment, with GRU model (case 3) having the shortest training. It may be due to the simplicity provided by GRU compared to LSTM, which makes the result of the vanilla RNN model surprising as it should be the simplest of all four.

The result in the case of the depth of the model is quite as expected. The shallower model gives a shorter training time with an optimal value of one recurrent layer and two dense layers. The capacity of one dense layer model may not be enough to learn abstract features of the data, causing it to behave slower. The result is the opposite in the width case, where a wider model gives a shorter time. The computation of units (neurons or cells) in a layer is done in parallel. Thus, more units do not contribute to the training time, causing a wider network to be more effective.

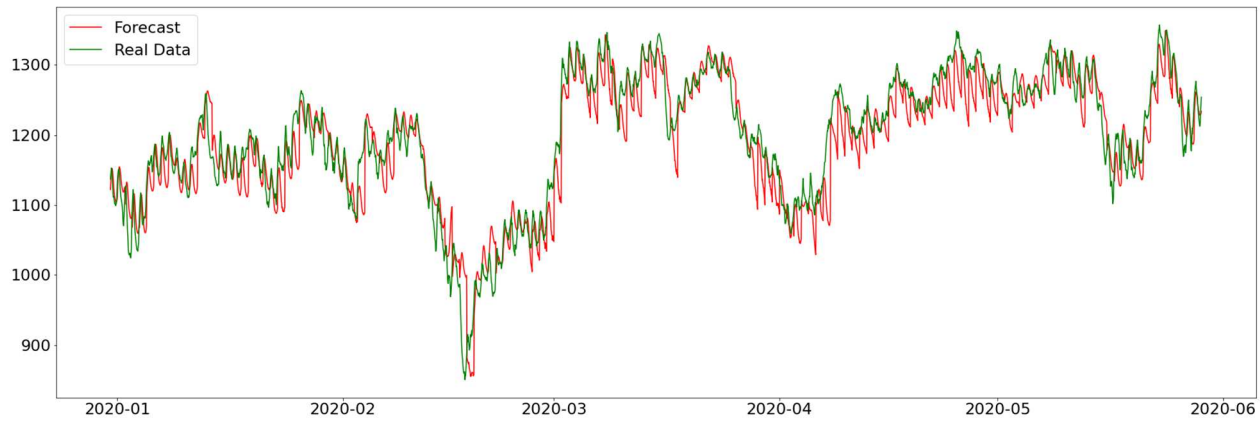

Fig. 5 Comparison of forecasted pressure profile with the real data in one semester period

For the last experiment, the training times of learning rate are similar except for the first case (lr=0.005) due to its unstable behavior. Even though case 3 gives the shortest time, we still can say that the best case is case 2 because the difference is only a matter of a second.

D. Forecasting result

In this last subsection, we review the forecasting result of the best model. Validation data consists of 18 months datapoint. For ease of observation, we take only six months for testing, i.e., data from January to June 2020. The forecasted profile is shown in Fig.5. It is shown in the figure that the forecasted profile is close to the real data. The trends, even though they fluctuated quite rapidly, are followed properly by the forecast. Some sharp spikes are shown to cause overshooting in value. However, the general trend is sufficiently captured.

For a closer look, we then take two date samples and forecast the pressure profile of the upcoming day with the input of the pressure profile of the given and previous day. The dates taken are 18th of April and 27th of July in the year 2020. The results are shown in Fig. 6. In the first case, the real pressure profile tends to decrease a little bit with some small fluctuations. The decrease is possibly caused by the drastic decrease in flowrate in Source 3 and Source 4. The model captures this decreasing trend, but the value is overshot, giving a drastic decrease. In the second case, the shortfall in Source 3 causes the next day pressure to fall in a short time before it increases again. This trend is captured perfectly by the model giving close profiles.

In field practice, the overshooting issue is tolerable because the predicted trend is needed most, whether the pressure will be decreased or increased in the day ahead. The error caused by overshooting is also not significant if we consider the scale of the value. In Fig.6(a), for instance, the average value difference is around 50 psig, which is small compared to the scale of 1000 psig. To see this in quantitatively way, we can also calculate the RE of the model. In the test case of Fig.6, the first case has RE around 3.284 % and the second case has RE around 1.023 %. If we test the model to the whole validation set, the RE obtained is around 2.232%. These values are sufficiently small compared to 5% commonly used maximum acceptable error in engineering.

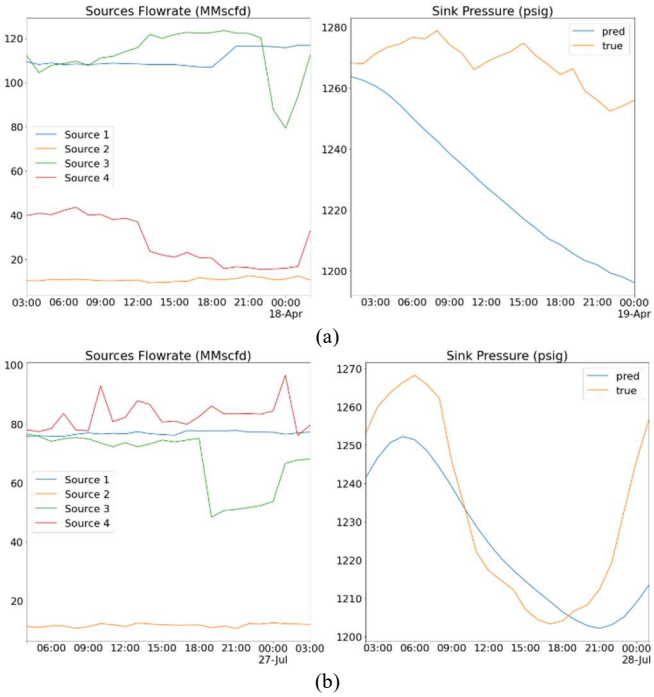

Fig. 6 Forecasting results

IV. CONCLUSION

The deep learning model, or specifically the recurrent neural network model, has been successfully implemented to forecast future pressure values in receiving facility of a natural gas transmission network given the historical flow rate data from all network end-nodes. The sensitivity of some model architecture and learning parameter variables is experimented with and analyzed. Overall validation MAE in almost all cases is roughly similar, around 0.37 but with significant differences of training time. The forecast profile agrees with the real data with a small acceptable error around the value of 2%.

ACKNOWLEDGMENT

This research is funded by Institut Teknologi Bandung (ITB) research grant.

REFERENCES

[1] T. Kiuchi, "An implicit method for transient gas flows in pipe networks," *Int J Heat Fluid Flow*, vol. 15, no. 5, pp. 378–383, 1994, doi: 10.1016/0142-727X(94)90051-5.

[2] C. Li, H. Wang, H. Yue, and S. Guo, "Fast difference scheme for the reaction-diffusion-advection equation with exact artificial boundary conditions," *Applied Numerical Mathematics*, vol. 173, no. 2018, pp. 395–417, 2022, doi: 10.1016/j.apnum.2021.12.013.

[3] Q. Guo, Y. Liu, Y. Yang, T. Song, and S. Wang, "Improved Adaptive Time Step Method for Natural Gas Pipeline Transient Simulation," *Energies (Basel)*, vol. 15, no. 14, p. 4961, Jul. 2022, doi: 10.3390/en15144961.

[4] P. Wang, S. Ao, B. Yu, D. Han, and Y. Xiang, "An efficiently decoupled implicit method for complex natural gas pipeline network simulation," *Energies (Basel)*, vol. 12, no. 8, 2019, doi: 10.3390/en12081516.

[5] P. Bangert, "Introduction to Machine Learning in the Oil and Gas Industry," in *Machine Learning and Data Science in the Oil and Gas Industry*, Elsevier, 2021, pp. 69–81. doi: 10.1016/B978-0-12-820714-7.00004-2.

[6] K. M. Hanga and Y. Kovalchuk, "Machine learning and multi-agent systems in oil and gas industry applications: A survey," *Comput Sci Rev*, vol. 34, p. 100191, 2019, doi: 10.1016/j.cosrev.2019.08.002.

[7] R. N. Anderson *et al.*, "Petroleum analytics learning machine to forecast production in the wet gas marcellus shale," *SPE/AAPG/SEG Unconventional Resources Technology Conference 2016*, pp. 132–147, 2016, doi: 10.15530/urtec-2016-2426612.

[8] Q. Cao, R. Banerjee, S. Gupta, J. Li, W. Zhou, and B. Jeyachandra, "Data driven production forecasting using machine learning," *Society of Petroleum Engineers - SPE Argentina Exploration and Production of Unconventional Resources Symposium*, 2016, doi: 10.2118/180984-ms.

[9] S. A. Sharaf, P. Bangert, M. Fardan, K. Alqassab, M. Abubakr, and M. Ahmed, "Beam pump dynamometer card classification using machine learning," *SPE Middle East Oil and Gas Show and Conference, MEOS, Proceedings*, vol. 2019-March, 2019, doi: 10.2118/194949-ms.

[10] U. Sinha, B. Dindoruk, and M. Soliman, "Machine learning augmented dead oil viscosity model for all oil types," *J Pet Sci Eng*, vol. 195, no. July, p. 107603, 2020, doi: 10.1016/j.petrol.2020.107603.

[11] U. T. Tygesen, K. Worden, T. Rogers, G. Manson, and E. J. Cross, "State-of-the-art and future directions for predictive modelling of offshore structure dynamics using machine learning," *Conference Proceedings of the Society for Experimental Mechanics Series*, vol. 2, pp. 223–233, 2019, doi: 10.1007/978-3-319-74421-6_30.

[12] K. Patel and R. Patwardhan, "Machine learning in oil & gas industry: A novel application of clustering for oilfield advanced process control," *SPE Middle East Oil and Gas Show and Conference, MEOS, Proceedings*, vol. 2019-March, 2019, doi: 10.2118/194827-ms.

[13] J. Zenisek, F. Holzinger, and M. Affenzeller, "Machine learning based concept drift detection for predictive maintenance," *Comput Ind Eng*, vol. 137, no. August, p. 106031, 2019, doi: 10.1016/j.cie.2019.106031.

[14] R. Pelta, N. Carmon, and E. Ben-Dor, "A machine learning approach to detect crude oil contamination in a real scenario using hyperspectral remote sensing," *International Journal of Applied Earth Observation and Geoinformation*, vol. 82, no. November 2018, p. 101901, 2019, doi: 10.1016/j.jag.2019.101901.

[15] F. I. Syed, M. Alshamsi, A. K. Dahaghi, and S. Neghabhan, "Artificial lift system optimization using machine learning applications," *Petroleum*, vol. 8, no. 2, pp. 219–226, Jun. 2022, doi: 10.1016/j.petlm.2020.08.003.

[16] M. R. Khan, S. Alnuaim, Z. Tariq, and A. Abdulraheem, "Machine learning application for oil rate prediction in artificial gas lift wells," *SPE Middle East Oil and Gas Show and Conference, MEOS, Proceedings*, vol. 2019-March, 2019, doi: 10.2118/194713-ms.

[17] G. Cui, Q. S. Jia, X. Guan, and Q. Liu, "Data-driven computation of natural gas pipeline network hydraulics," *Results in Control and Optimization*, vol. 1, no. December, p. 100004, 2020, doi: 10.1016/j.rico.2020.100004.

[18] N. Elshaboury, A. Al-Sakkaf, G. Alfalah, and E. M. Abdelkader, "Data-Driven Models for Forecasting Failure Modes in Oil and Gas Pipes," *Processes*, vol. 10, no. 2, pp. 1–17, 2022, doi: 10.3390/pr10020400.

[19] X. Li, J. Wang, and G. Chen, "A machine learning methodology for probabilistic risk assessment of process operations: A case of subsea gas pipeline leak accidents," *Process Safety and Environmental Protection*, vol. 165, pp. 959–968, Sep. 2022, doi: 10.1016/j.psep.2022.04.029.

[20] M. Giuliani *et al.*, "Flaring events prediction and prevention through advanced big data analytics and machine learning algorithms," *Offshore Mediterranean Conference and Exhibition 2019, OMC 2019*, no. April 2021, 2019.

[21] J. You *et al.*, "Assessment of enhanced oil recovery and CO2 storage capacity using machine learning and optimization framework," *Society of Petroleum Engineers - SPE Europec Featured at 81st EAGE Conference and Exhibition 2019*, vol. i, 2019, doi: 10.2118/195490-ms.

[22] H. C. Phan and A. S. Dhar, "Predicting pipeline burst pressures with machine learning models," *International Journal of Pressure Vessels and Piping*, vol. 191, no. March, 2021, doi: 10.1016/j.ijpvp.2021.104384.

[23] Z. Zhong, A. Y. Sun, Y. Wang, and B. Ren, "Predicting field production rates for waterflooding using a machine learning-based proxy model," *J Pet Sci Eng*, vol. 194, no. December 2019, p. 107574, 2020, doi: 10.1016/j.petrol.2020.107574.

[24] M. Petkovic, T. Koch, and J. Zittel, "Deep learning for spatio-temporal supply and demand forecasting in natural gas transmission networks," *Energy Sci Eng*, vol. 01, no. December, 2021, doi: 10.1002/ese3.932.

[25] T. Bikmukhametov and J. Jäschke, "Oil production monitoring using gradient boosting machine learning algorithm," *IFAC-PapersOnLine*, vol. 52, no. 1, pp. 514–519, 2019, doi: 10.1016/j.ifacol.2019.06.114.

[26] E. A. M. VÃlez, F. R. Consuegra, and C. A. B. Arias, "EOR Screening and Early Production Forecasting in Heavy Oil Fields: AMachine Learning Approach," *SPE Latin American and Caribbean Petroleum Engineering Conference Proceedings*, 2020, doi: 10.2118/199047-MS.

[27] Y. Zhang and S. Hamori, "Forecasting crude oil market crashes using machine learning technologies," *Energies (Basel)*, vol. 13, no. 10, 2020, doi: 10.3390/en13102440.

[28] Sousa, Ribeiro, Relvas, and Barbosa-Póvoa, "Using Machine Learning for Enhancing the Understanding of Bullwhip Effect in the Oil and Gas Industry," *Mach Learn Knowl Extr*, vol. 1, no. 3, pp. 994–1012, 2019, doi: 10.3390/make1030057.

[29] Y. Bengio, A. Courville, and I. J. Goodfellow, *Deep Learning*. 2016. doi: 10.2172/1462436.

[30] Y. Yu, X. Si, C. Hu, and J. Zhang, "A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures," *Neural Comput*, vol. 31, no. 7, pp. 1235–1270, Jul. 2019, doi: 10.1162/neco_a_01199.