Neural Computing and Applications (2020) 32:13639–13649
https://doi.org/10.1007/s00521-020-04773-2

ORIGINAL ARTICLE

Purities prediction in a manufacturing froth flotation plant: the deep learning techniques

Yuanyuan Pu[1,2] · Alicja Szmigiel[2] · Derek B. Apel[2]

Received: 26 November 2019 / Accepted: 3 February 2020 / Published online: 13 February 2020

Abstract
Accurate and timely investigation to concentrate grade and recovery is a premise of realizing automation control in a froth flotation process. This study seeks to use deep learning technologies modeling a manufacturing flotation process, forecasting the concentrate purities for iron and the waste silica. Considering the size and temporality of engineering data, we adopted a long short-term memory to form the core part of the deep learning model. To perform this process, 23 variables reflecting a flotation plant were monitored and collected hourly over a half year time span, then wrangled, split, and restructured for deep learning model use. A deep learning model encompassing a stacked long short-term memory architecture was designed, trained, and tested with prepared data. The model's performance on test data demonstrates the capability of our proposed model to predict real-time concentrate purities for iron and silica. Compared with a traditional machine model typified by a random forest model in this study, the proposed deep learning model is significantly more competent to model a manufacturing froth flotation process. Expected to lay a foundation for realizing automation control of the flotation process, this study should encourage deep learning in mineral processing engineering.

Keywords Froth flotation · Deep learning · Long short-term memory · Concentrate purity

1 Introduction

Froth flotation is an extensively used physiochemical mineral processing technology for separating particles of valuable and unwanted minerals, with the fact that different minerals have a different physicochemical surface characteristic, either hydrophobic or hydrophilic [1, 2]. Timely investigation to concentrate grade and recovery with online monitoring or estimation in a froth flotation plant is important to processing engineers, which is fundamental to process control and optimization. Online monitoring requires purchasing and maintaining costly and sophisticated instruments, which necessitates development of models for concentrate grade and recovery estimation based on processing parameters. Some difficulties, such as a lack of comprehensive knowledge of the physicochemical rules for flotation subprocess and the need of deducing and solving mathematic equations for a moving boundary (collection/froth zone interface), make it impractical to develop traditional knowledge-based models which usually are explicit calculations for flotation process [3, 4]. Thus, data-driven methods that require less prior knowledge about the nature of the relationship between the input/output variables have been an alternative strategy for modeling the flotation process.

Compelled only by acquired data, rather than by intuition or personal expertise, a data-driven model development for the flotation process encompasses the following steps: (1) Collect sufficient and high-quality data samples based on the control factors determined to influence a flotation process. (2) Wrangle data for training preparation and split data into training and validation sets. (3) Choose a suitable model type and implement model training. (4)

Yuanyuan Pu and Alicja Szmigiel have contributed equally to this work.

✉ Yuanyuan Pu
yp@ualberta.ca

✉ Derek B. Apel
derek.apel@ualberta.ca

[1] College of Resources and Safety Engineering, State Key Laboratory of Coal Mine Disaster Dynamics and Control, Chongqing University, Chongqing 400044, People's Republic of China

[2] School of Mining and Petroleum Engineering, University of Alberta, Edmonton, AB T6G 2R3, Canada

Evaluate the trained model using unseen validation data and tune the hyperparameters to improve model performance. (5) Deploy model for engineering practice.

As a major instrument of data-driven strategy, machine learning has demonstrated its success in many data-intensive areas such as spam filtering, machine translation, and natural language processing. Spontaneously, machine learning has sparked the interest of engineers in mineral processing [5–8]. Various machine learning models have been developed for modeling flotation processes including multilayer perception [1, 3, 9], support vector machine [10–12], and random forest [13] as well as gaining some achievements in modeling laboratory flotation processes which only have limited, simple process data. However, shallow machine learning models or statistic learning models have turned out to perform poorly for those tasks encompassing massive data and complicated data structures such as high-dimensional, auto-correlated, and temporal, meaning that the aforementioned shallow machine learning models are not competent for modeling flotation processes in engineering. By contrast, deep learning models typified by the convolutional neural network (CNN), the recurrent neural network (RNN), and the generative adversarial network (GAN) have proven to be ideal tools tackling large-scale and complicated datasets on account of their deep, intricate and flexible architectures [14]. Currently, only a few successful cases that employ deep learning in mineral processing are reported but not especially for manufacturing froth flotation [15].

To overcome the lack of deep learning applications in manufacturing froth flotation as well as to realize a dynamic prediction for output purities, this study profits from the development of an evolved RNN—the long short-term memory (LSTM) in tackling time-series prediction tasks. We construct a stacked LSTM architecture to model an engineering froth flotation process with the goal of predicting the iron and silica purities based on several fast, easy measurements from a manufacturing froth flotation plant. This plant employs reverse cationic flotation to separate iron from undesired particles that are mainly silica. Unlike previous studies that usually used limited laboratory flotation data to train machine learning models, this study collected more than 3900 hourly based manufacturing data samples (time span nearly a half year) to train an LSTM model. The whole flow of data munging and restructuring, model design, and training and result interpretation are exhibited in this study to demonstrate the feasibility of using a deep learning model in engineering flotation process modeling. The rest of this paper is organized as follows: Section 2 introduces the engineering background and database used for our paper; Sect. 3 exhibits the structure of LSTM and its logistics; Sect. 4 provides the results and discussion; Sect. 5 delivers the conclusions.

2 Engineering background and database information

Reverse cationic flotation is the most commonly used process to separate iron from undesired particles such as silica, which is the dominant gangue mineral in iron ores. In this process, iron minerals remain in the water and create sediment with a high concentration of iron; At the same time, silica particles attach to air bubbles and float to the surface [16].

From a real manufacturing flotation plant, the provided dataset in this paper contains information about the process of separating silica particles from iron ore using reverse cationic flotation. Figure 1 exhibits the flowsheet of the iron ore flotation process with zoom on a single flotation cell in our target flotation plant. In this flotation plant, iron pulp (about 400 *t*/*h*) is being fed into three flotation cells that run in parallel (flotation cells 1, 2, and 3 on Fig. 1). Froth with high concentration of SiO_2 is constantly removed from the surface. Depressed sediment with a high amount of iron flows into the next cell to repeat the same process (flotation cell 4, 5, 6 in Fig. 1). Output streams from flotation cells 4, 5, 6 create an input stream to the final flotation (flotation cell 7 in Fig. 1).

From 12 p.m. on March 29, 2017, to 23 p.m. on September 9, 2017, 3948 time-continuous data were sampled on an hourly base. The first five data are shown in Table 1 (each data sample is in one column) as a snapshot. Each data sample contains 23 measurable values that can be categorized into four types: raw materials (row 2 to row 3) referring to the purities of iron and silica fed into the plant; environment variables (row 4 to row 8); process variables (row 9 to row 22); and processed materials (row 23 to row 24). In this paper, our goal is to predict purity of processed materials at time t with provided measurable values.

Figure 2 exhibits the purities of the materials before and after processing during the whole monitoring period. As Fig. 2 shows, the goal of flotation is to improve the purity of iron, along with reducing the purity of silica. The output purities of materials are not only governed by feed purities but also impacted by environment parameters and process variables because during some periods (for example, May 13 to June 13) the feed purities for iron and silica were constant but the output purities varied. Even though the feed purities for iron and silica fluctuated significantly, the output purities were relatively stable, which suggests the temporality of output purities.

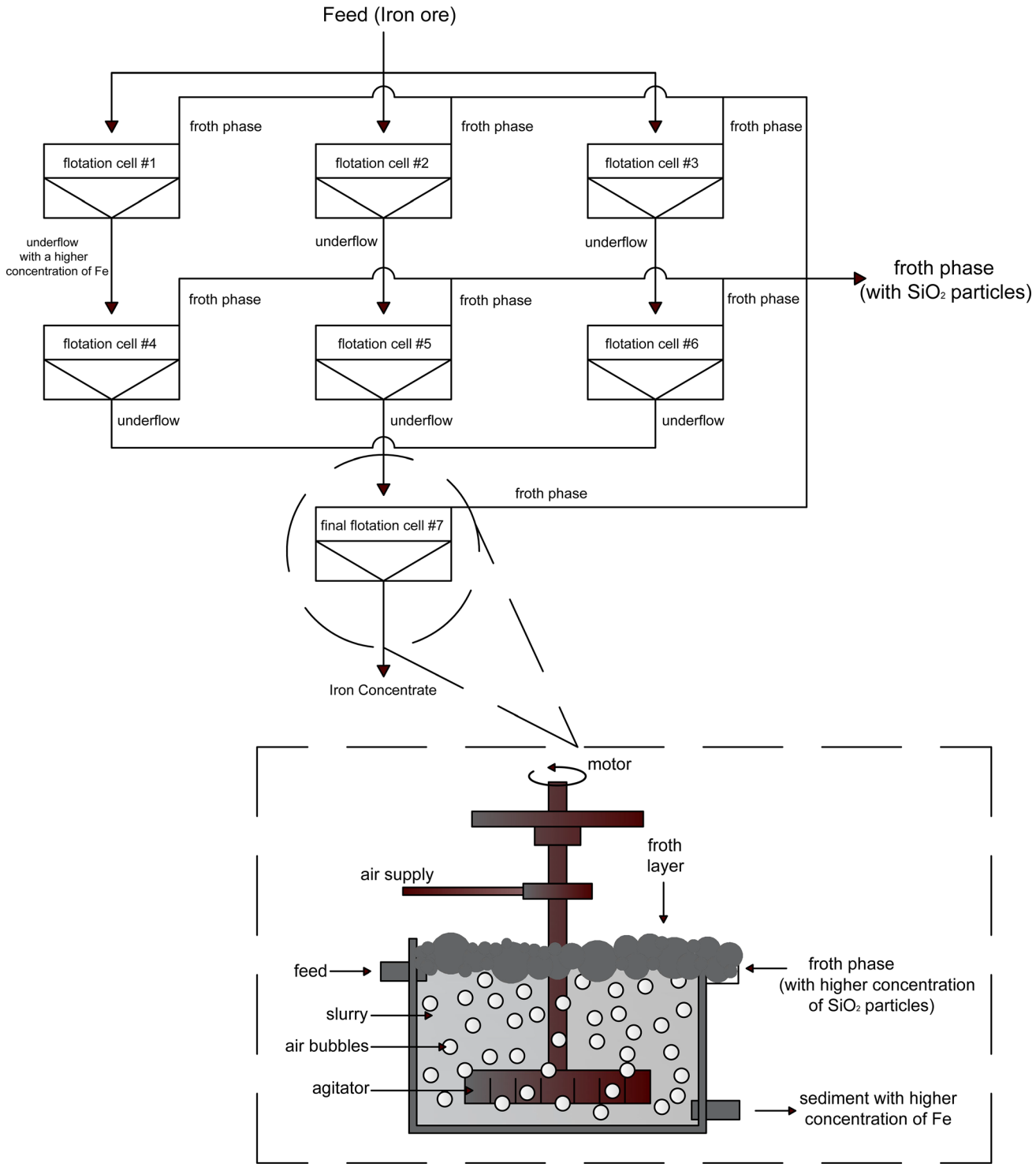

Fig. 1 Flowsheet of iron ore flotation process with zoom on single flotation cell

Figure 3 provides a similar view for data temporality, which exhibits scatter plots for both iron and silica concentrates, with y-axis showing observations at time $t + 1$ and x-axis showing observations at time t (1 h lag). Both points cluster along a diagonal line from the bottom-left to the top-right, although this trend for the iron concentrate is more apparent. Both subfigures suggest a positive correlation between material concentrates at time t and $t + 1$; in other words, the data temporality. This is the main consideration of employing LSTM in this paper: We treat these data not only as cross-sectional samples but also as time-series data flow. The output purities at time t are not only

Table 1 A snapshot of the collected engineering database

Date	3/29/2017 12:00	3/29/2017 13:00	3/29/2017 14:00	3/29/2017 15:00	3/29/2017 16:00
Input values					
Raw material					
%Iron feed	60.18	60.18	60.18	60.18	59.54
%Silica feed	9.34	9.34	9.34	9.34	9.56
Environment variables					
Starch flow	1060.86	2034.93	1435.43	618.46	1367.50
Amina flow	379.84	322.23	474.66	396.38	317.13
Ore pulp flow	400.98	400.47	399.16	398.94	400.84
Ore pulp pH	9.53	9.70	9.69	9.85	9.94
Ore pulp density	1.55	1.53	1.65	1.56	1.53
Process variables					
Flotation column 01 air flow	200.04	199.99	200.02	199.94	199.88
Flotation column 02 air flow	195.58	195.11	195.60	195.66	196.16
Flotation column 03 air flow	199.98	199.77	199.93	200.04	199.91
Flotation column 04 air flow	295.10	295.10	295.10	295.10	295.10
Flotation column 05 air flow	306.40	306.40	306.40	306.40	306.40
Flotation column 06 air flow	250.07	250.03	249.98	249.98	250.06
Flotation column 07 air flow	249.99	250.06	250.11	250.03	250.12
Flotation column 01 level	753.72	848.68	851.95	855.91	851.60
Flotation column 02 level	726.56	777.78	776.29	780.37	784.45
Flotation column 03 level	860.56	869.42	879.70	882.09	884.84
Flotation column 04 level	477.35	483.82	456.15	449.41	450.12
Flotation column 05 level	452.51	469.45	453.23	448.58	451.77
Flotation column 06 level	478.22	471.61	447.71	450.70	451.57
Flotation column 07 level	470.11	462.67	453.48	448.66	449.41
Output values					
Processed materials					
%Iron concentrate	66.44	66.57	66.64	66.41	63.63
%Silica concentrate	1.36	1.43	1.33	1.27	5.50

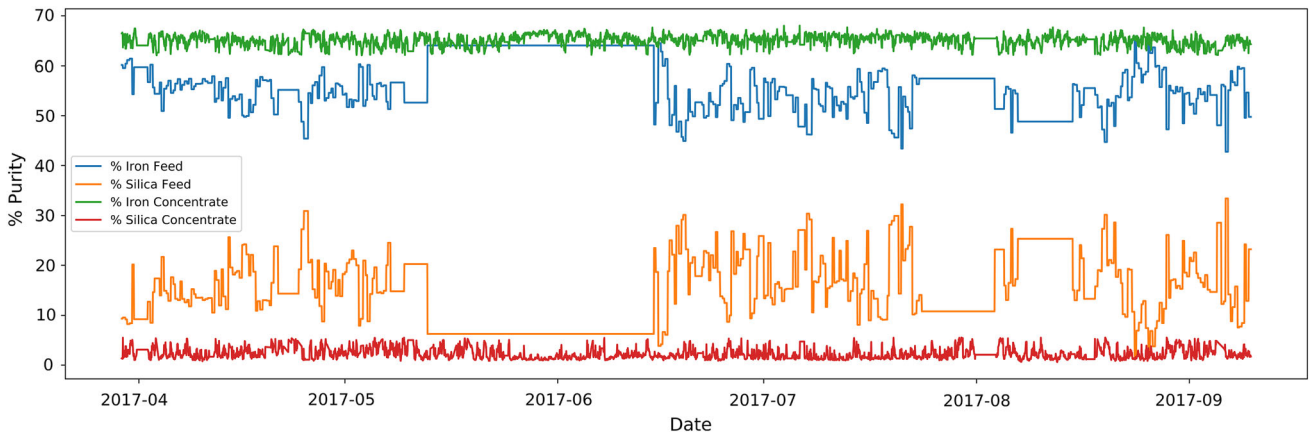

Fig. 2 Comparisons of concentrate purities before and after flotation process

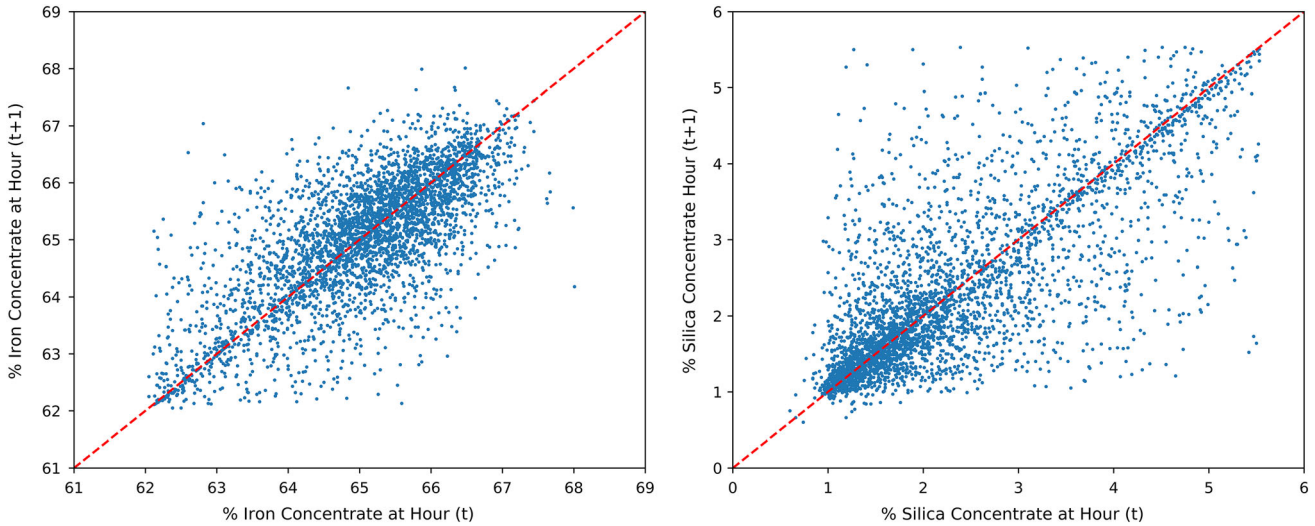

Fig. 3 Time correlation for concentrate purities

impacted by the input parameters at time $t - 1$ but also correlated to output purities at a certain previous period (time lags).

3 Model construction

3.1 Introduction of LSTM

The feedforward neural network (FNN) represented by the multilayer perceptron is ideally suitable for cross-sectional data samples but badly performed for time corrected data since information flows only in one direction in FNN architecture. Compared with FNN, recurrent neural network (RNN) specializes in tackling time-series data due to its special architecture [17]. Topologically, the neurons in an RNN are self-connected along a temporal sequence. RNN has information traveling in both directions by importing the loops in the architecture. The intermediate computations (memory) from preceding inputs are fed back into the network, deriving the final outputs. The formulized information flow in a vanilla RNN is shown as formula (1), where x_t, y_t, and h_t are input, output, and hidden state vectors at time t; W, U, and b are trainable parameters in the network; σ_h and σ_y are activation functions.

$$\begin{aligned} h_t &= \sigma_h(W_h x_t + U_h h_{t-1} + b_h) \\ y_t &= \sigma_y(W_y h_t + b_y) \end{aligned} \tag{1}$$

Theoretically, a vanilla RNN is capable of exhibiting the temporal dynamic behavior by keeping track of arbitrary long-term dependencies in the input sequences. Nevertheless, computationally, a vanilla RNN may encounter gradient vanishing or exploding in the backpropagation training process, possibly hindering a vanilla RNN from having a long-term memory for input sequences from the early phase. A special RNN architecture, the LSTM, is developed to overcome gradient vanishing and exploding by importing gate mechanism into a vanilla RNN [18]. In addition to the input vector x_t and the hidden state vector h_t, LSTM introduces a new input c_{t-1} reflecting the internal state at time t to establish the gating mechanism. Formula 2 provides formulized information flow in an LSTM, wherein f_t, i_t, o_t represent the forget gate, input gate and output gate, respectively, $\tilde{c}_t$ stands for a candidate state, and '$\otimes$' is a Hardmard product for vectors.

$$\begin{aligned} \begin{bmatrix} \tilde{c}_t \\ o_t \\ i_t \\ f_t \end{bmatrix} &= \begin{bmatrix} \tanh \\ \sigma \\ \sigma \\ \sigma \end{bmatrix} \left(W \begin{bmatrix} x_t \\ h_{t-1} \end{bmatrix} + b \right) \\ c_t &= f_t \otimes c_{t-1} + i_t \otimes \tilde{c}_t \\ h_t &= o_t \otimes \tanh(c_t) \end{aligned} \tag{2}$$

Figure 4 exhibits an evolution process from FNN to LSTM. In this study, we considered the correlation between material purities at the current moment and preceding hours; at the same time, we wanted to ensure that our model is capable of revealing the impact of previous outputs to current observations (a "long-term memory"). LSTM is an ideal choice to model the intrinsic relationship of our collected data.

3.2 Training data preparation for LSTM

LSTM requires the features of each training sample restructured as a two-dimensional vector (*timesteps*, *inputs*). Timesteps relate to the length of retrospective time

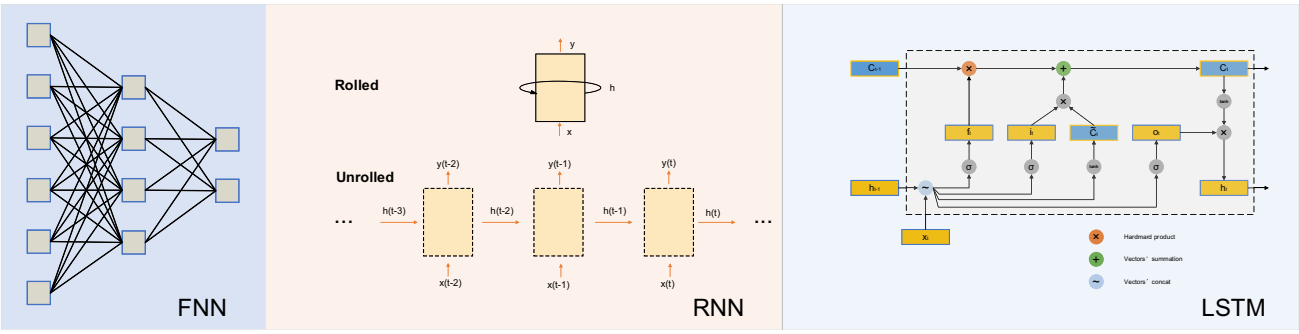

Fig. 4 Basic architectures for FNN, RNN, and LSTM

duration that is determined to impact current outputs. For example, timesteps T (hours) mean that the outputs at time t are corrected to the outputs at time $t-1$, $t-2$, …, $t-T$. Timesteps are a hyperparameter in LSTM model, which is determined by the user's expertise. Larger timesteps may consider temporal relation more complete but would enlarge the size of each training sample, burdening computation. In this study, we consider timesteps as 6 h for LSTM based on the industry experience of this manufacturing plant. For inputs determination, as we discussed in Sect. 2, output purities at time t should be linked to input values at time $t-1$, $t-2$, …, $t-T$ as well as the output purities at these moments. The label for training samples is the purities of iron and silica, a one-dimensional vector (% iron concentrate, % silica concentrate). Figure 5 explains the construction of the first three training samples for timesteps of 6 h. Following this rule of data restructure, we produced 3942 training samples, of which the first 70% samples (2760) were used for model training and the remaining 30% (1182) were used as model testing. Accordingly, the time range for training samples is between 12 p.m., March 29, and 3 a.m., July 23, and for testing samples is between 3 a.m., July 23, and 23 p.m., September 9. Before we fed data into the model for training, all data were scaled in the range [0, 1] using formula (3) to eliminate data range varying.

$$x_{\text{new}} = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{3}$$

3.3 Model building and training

This study employs Keras, a widely used Python deep learning library, to construct and train our LSTM model. Although a large-scale single-layer network can approximate all continuous functions [19], we adopted a stacked LSTM architecture, referring to stacking multiple LSTM hidden layers, rather than a single-layer LSTM in this study

Fig. 5 Training sample construction for LSTM

for two reasons. A deep architecture allows hidden states at each level to operate at different timescales [20]. Also, increasing the depth of the network requires a smaller number of neurons (cells) for each layer, hence shortening the training time. Although some scholars proposed novel algorithms for hyperparameters configuration [15], there is no a universal way setting up hyperparameters including the number of layers and memory cell numbers of each layer in an LSTM, which allows many time-consuming heuristic methods to take over. This study focuses on investigating the feasibility of using deep learning in a time-series task, thereby not paying excessive attention to model hyperparameter tuning. Our model is comprised of three LSTM layers with 30 memory cells in each layer, followed by a fully connected layer at the tail for generating outputs. Figure 6 and Table 2 show the model structure and parameter numbers for different timesteps.

For model training, the mini-batch gradient descent algorithm was used to optimize all trainable parameters. The batch size was set as 32, meaning that parameters were updated every 32 training samples. In practice, an improved mini-batch gradient descent algorithm, adam [21], was adopted in the training process with a learning rate of 0.001 to overcome some shortcomings resulting from using a vanilla mini-batch gradient descent algorithm. In order to accelerate model training, we employed CUDA, a GPU parallel computing platform, in the training process. We produce Fig. 7 showing train and validation loss in training process. Both losses decrease and converge each other after 60 training epoch, demonstrating the proposed model a good fit.

4 Results and discussion

The training process includes 100 epochs, lasting approximately 100 s on a PC with a Core i7 8700 CPU, GeForce RTX 2070 GPU, and 16 GB RAM. Then, 1182 testing samples were fed into the trained model to predict material concentrates. Figure 8 shows the pointwise comparisons of

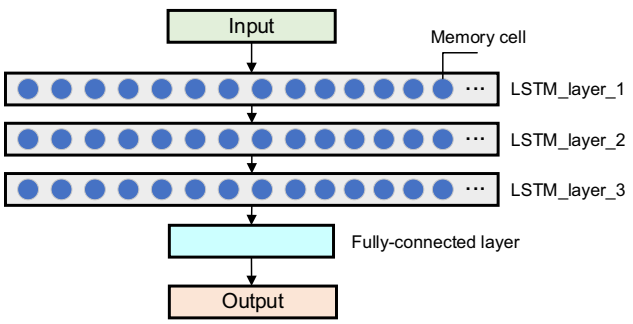

Fig. 6 Proposed deep learning architecture

Table 2 Model summary for proposed deep learning model

Layer	Output shape	Param #	Summary
LSTM_1	(6, 30)	6600	Total params: 21,542
LSTM_2	(6, 30)	7440	Trainable params: 21,542
LSTM_3	(6, 30)	7440	Non-trainable params: 0
FC layer	(1, 2)	62	

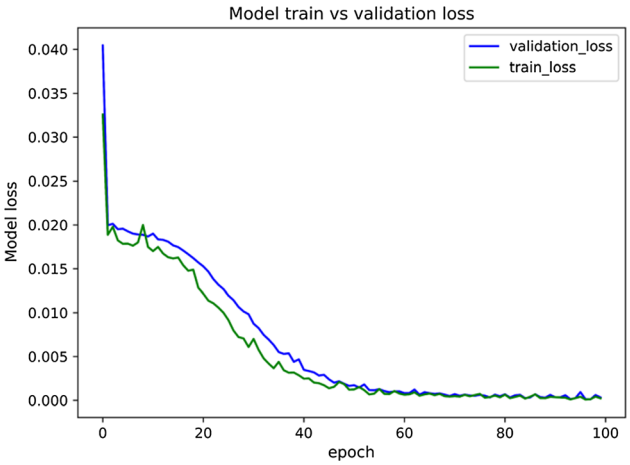

Fig. 7 Train and validation loss in model training

real material concentrate values collected from the flotation plant and the predicted values generated by LSTM. Both predicted values for silica and iron come close to the real values, located in the 95% confidence interval. The R-square values for predicted silica and iron purity are 0.77 and 0.71, respectively. Moreover, the goal of mineral processing is to increase iron purity, while decreasing silica purity, characterized by an opposite fluctuating trend of real silica and iron purities (orange lines). Our predicted values also follow this fluctuating trend, when predicted iron purity is increasing, predicted silica purity is decreasing, and vice versa. Crests and troughs on predicted value curves for silica and iron are reciprocally corresponded precisely. A couple of minor imperfections appear in the time range when the real outputs are constant. For example, between 8 p.m., July 31, and 9 p.m., August 3 (test sample from #226 to #299), the monitored purities for iron and silica are constant (65.44% for iron and 2.08% for silica). But our model gives fluctuated predictions for this time range, even though predictions are still contained in the confidence interval.

In addition, we produced scatter plots for further investigation of predicted results. The x-axis represents the real values, whereas the y-axis represents predicted values. Theoretically, the perfect predicted samples should locate on the diagonal line (red dashed line), which has the same

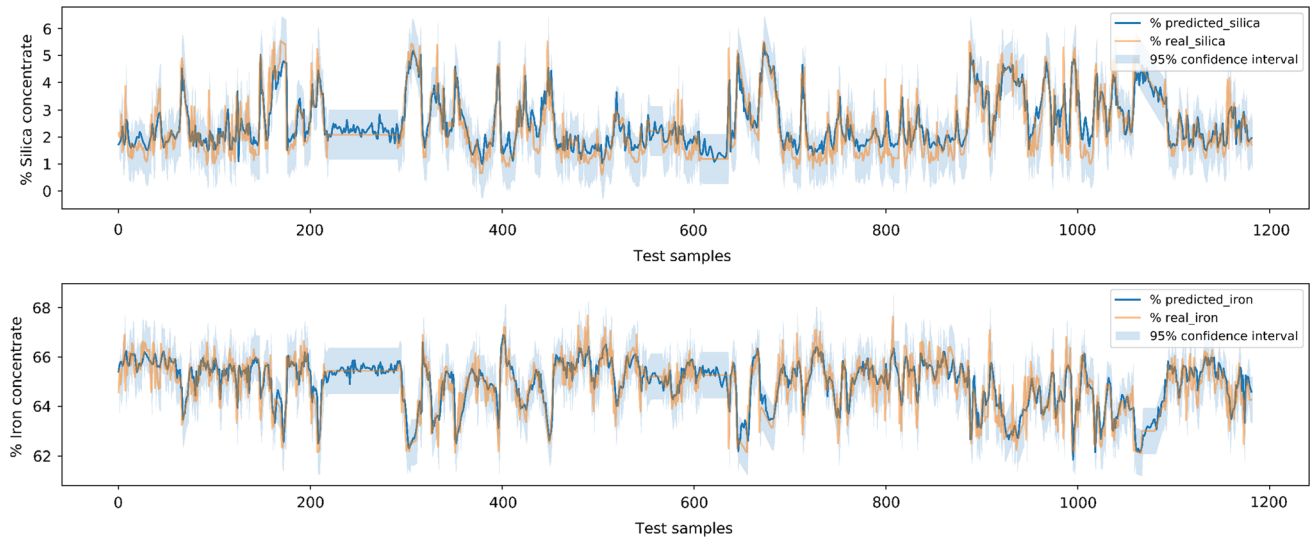

Fig. 8 A comparison of real concentrate purities and predicted purities

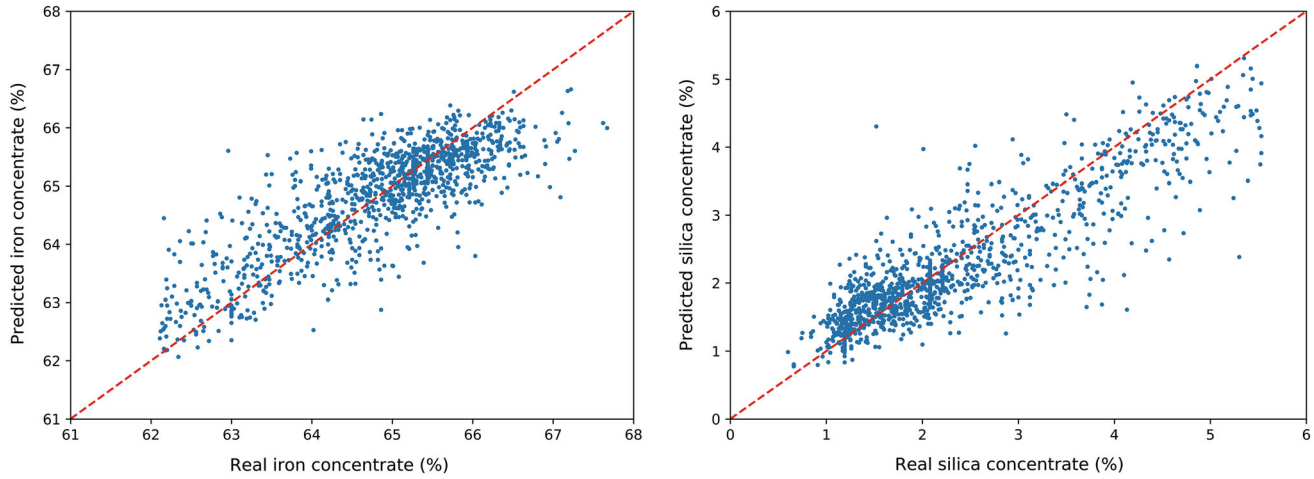

Fig. 9 Scatter plots for real purities and predicted purities

predicted value and real value. As Fig. 9 shows, most points cluster in the vicinity of the diagonal lines, demonstrating a good performance for the deep learning model. Over the 1182 testing samples, 692 predicted values for iron purity are higher than real values while 489 predicted values are lower than real values. By contrast, for silica purity, 634 predicted values are higher and 548 predicted values are lower. The model shows a slight bias toward larger predicted values for both iron and silica, but not significantly.

The predicted results achieve *R*-square value 0.77 for silica and 0.71 for iron, demonstrating the success of the proposed LSTM model. However, the performance of the proposed model still has room for improvement regarding the following issues: First, collecting more training samples would be the most straightforward and effective way to boost the model performance. Empirically, the number of training features and the number of training samples should be positively correlated [22]. Considering the large number of training features (23) in our task, the larger number of training samples is helpful to improve the model's interpretability. The second way to enhance model performance is to optimize hyperparameters, including the number of nodes and layers in the model, the type of activation function, the batch size. Implementing an exhaustive search is able to find a tuple of hyperparameters yielding an optimal model but at the same time burdening the computation. In this study, we preset hyperparameters intuitively without conducting hyperparameters tuning due to the limited computation resources. An easy-to-ignore parameter is the sequence length in LSTM, reflecting the time window impacting the model outputs. There is no rule of thumb to determine the sequence length as it totally depends on the nature of the input data and the inner

correlations. In this study, we empirically chose the sequence length as 6 h. But this number can be altered by implementing a validation process, comparing the prediction accuracies of models with different sequence lengths.

The proposed LSTM model helps engineers foresee iron and silica purities when the raw materials are fed into the plant. In the case that predicted material purities deviate from the expected values, engineers need adjust the input values to achieve the anticipated values. In order to determine which input value should be altered first, we must understand the impact of each input value on material purities, meaning to investigate the importance for each input value. The proposed method is permutation feature importance by calculating the change of the model prediction error when a single feature value is randomly shuffled. Especially, by permuting a single feature value, the model prediction error is expected to increase if the model relies on this feature for prediction. Instead, an unimportant feature leaves the model prediction error almost unchanged, even though the value is shuffled. Figure 10 exhibits the importance for 23 input features. Following this bar chart, if the gap between predicted material purities and expected material purities is wide, altering values of feature with greater importance, such as "Ore Pulp pH," "Amina Flow," and "Starch Flow," is recommended. If the gap is narrow, adjusting values of features with little impact on outputs, such as several airflow inputs, is recommended. Also, this chart tells us that the iron feed and silica feed are not the two most crucial elements influencing the output purities. A part of environment parameters influence the output purities more significantly.

LSTM considers the temporal correlation between training samples, treating training data as many sequential samples. In order to understand the rationality of considering temporal correlation of training data, we built another machine learning model, the random forest (RF) [23] that ignores the temporal correlation and treats training data as cross-sectional samples. The random forest is an ensemble model consisting of a multitude of decision trees, averaging the prediction of individual trees. In this study, we assigned the number of decision trees as 100. The random forest does not need a special structure for its training sample. To guarantee a fair competition of models, we still split 3948 training samples into the first 70% training samples and the remaining 30% test samples. It is worth noting that the dimension of the training feature for RF is (1, 21), which is different from that of LSTM (6, 23). Figure 11 shows the structure of training samples and the model architecture. After model training, we tested the trained RF on 1182 test samples that have the same test data domain as the LSTM model. Figure 12 exhibits the predicted results of RF. Obviously, most predicted values for both silica and iron heavily deviate from the true value, not located in the 95% confidence interval. The R-square value for silica is only 0.076. For iron, R-square is even less, only 0.014. All these proofs demonstrate that the proposed RF model fails to extract meaningful characteristics from the training data. In this study, the provided data, obtained by continuous monitoring from the froth flotation plant, have a natural temporal ordering. The RF model treats these data as cross-sectional samples, without considering the impact of previously observed values to the future values. By contrast, LSTM employs sequential data as model inputs, implying that data temporality has been absorbed in the model. This explains why LSTM performs much better than RF even

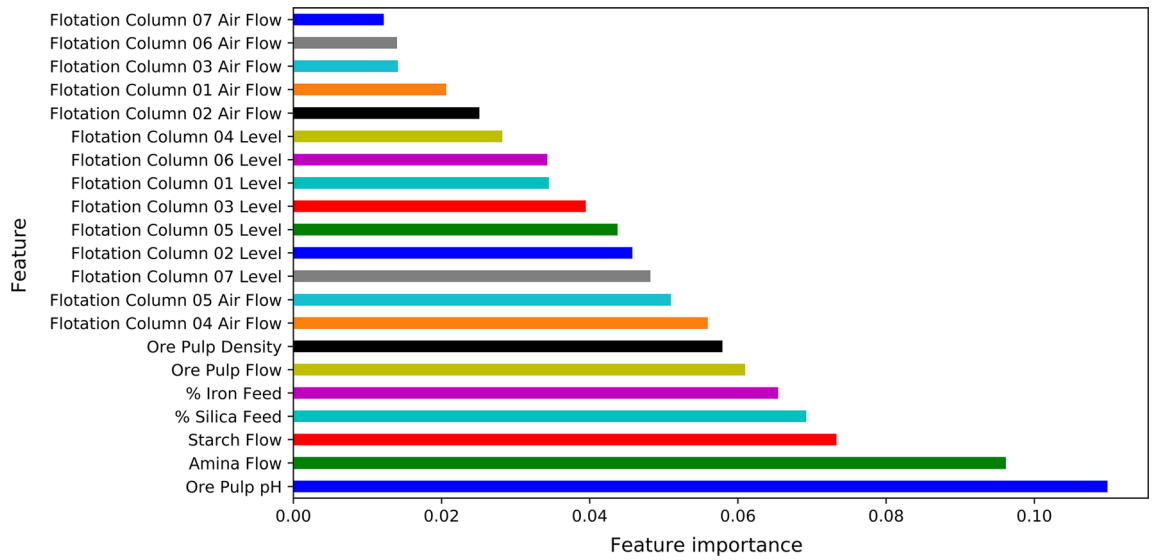

Fig. 10 Importance for features

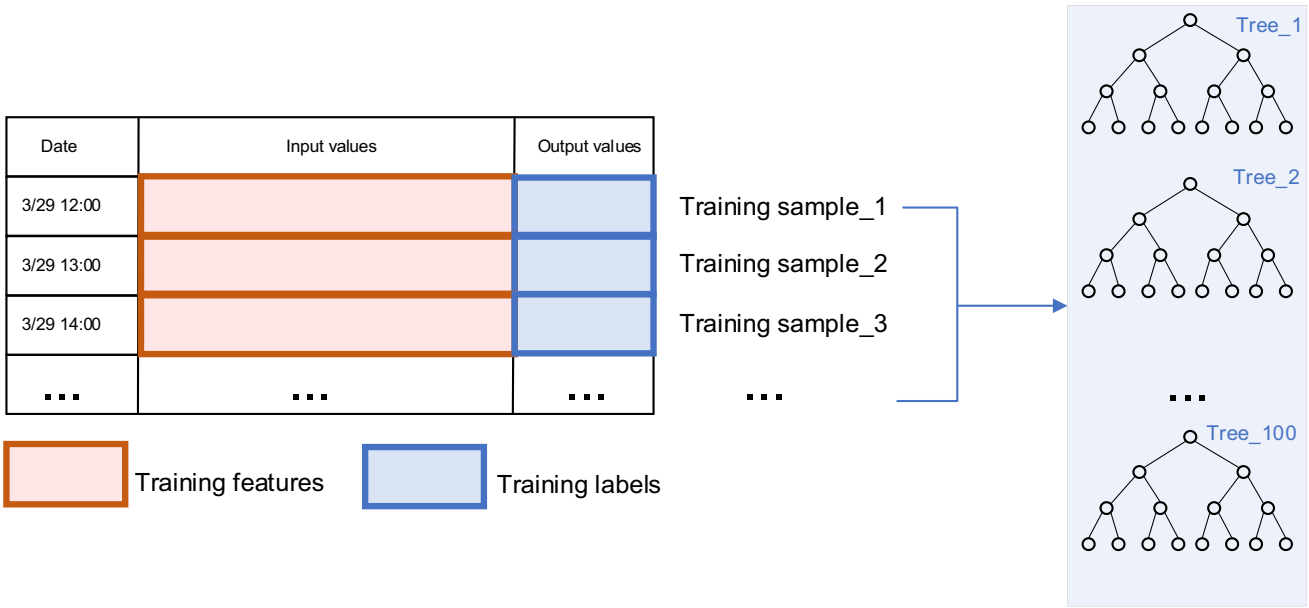

Fig. 11 Required data structure of RF and its architecture

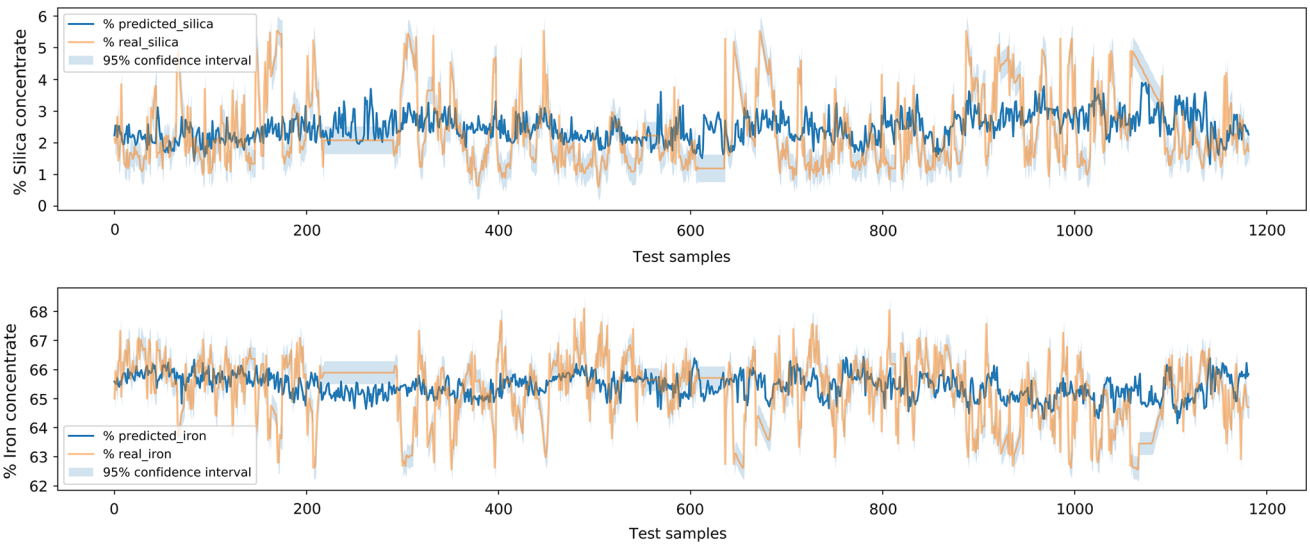

Fig. 12 Model performance of RF

when both models were trained and tested with the same data domain.

Despite the successful prediction for processed purities using LSTM in this study, there remain some problems that require further study. The first problem would be the "black-box" property of LSTM or even all deep learning models, referring to the lack of knowledge regarding how the output was arrived at. In other words, we cannot extract an explicit function representing the trained LSTM. Based on the factor importance investigation shown in Fig. 10, we expect to adjust the output purities by altering the input values. However, Fig. 10 cannot provide a quantitative relationship between input values and output values due to the lack of an explicit function. Thus, we do not know how much of the alternation for each input value can result in the expected output values. In future research, quantifying the significances of input variables should be an issue worthy of study. The second problem is related to sampling frequency. The optimal sampling frequency should be case by case, determined by the time span of the whole processing process. The longer time span requires reducing sampling frequency, while the shorter time span requires increasing sampling frequency. Usually however, a high sampling frequency means more data collected, which requires models that can handle massive data. In this study, we just built a model with already given data without a

detailed site investigation. In order to achieve better prediction accuracy, it is necessary to investigate the time span of the processing process before model construction.

5 Conclusion

To summarize, this paper developed an LSTM model in a manufacturing froth flotation plant to predict processed material purities based on measurable input values, exhibiting the training data preparation and restructuring, model building and training and results analysis. Different from vanilla machine learning models treating input data as cross-sectional samples, the proposed LSTM model can reveal the data temporality based on its distinct architecture, which logically delivers a better prediction accuracy than shallow models typified by a random forest in this study. In addition, we analyzed the factor importance to output purities for assisting in altering the input values in cases in which there is a gap between expected output purities and predicted output ones. Our proposed model also presents ways to further enhance the model performance. Results gained from this study provide a promising avenue to processing engineers of timely investigating processed material purities, which would contribute to productivity optimization and automatic processing control by combining hardware. The success of proposed LSTM model should serve to encourage deep learning methods in froth flotation process and mineral processing.

Acknowledgements This study was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) under Collaborative Research and Development (CRD) Grant (NSERC RGPIN-2019-04572). And also, supports from Chinese Scholarship Council were gratefully acknowledged.

Compliance with ethical standards

Conflict of interest There are no conflicts of interest to disclosure.

References

1. Jahedsaravani A et al (2016) Froth-based modeling and control of a batch flotation process. Int J Miner Process 146:90–96
2. Wills BA, Finch J (2015) Wills' mineral processing technology: an introduction to the practical aspects of ore treatment and mineral recovery. Butterworth-Heinemann, Oxford
3. Nakhaei F et al (2012) Recovery and grade accurate prediction of pilot plant flotation column concentrate: neural network and statistical techniques. Int J Miner Process 110:140–154
4. Vieira S, Sousa J, Durão F (2005) Fuzzy modelling strategies applied to a column flotation process. Miner Eng 18(7):725–729
5. McCoy J, Auret L (2019) Machine learning applications in minerals processing: a review. Miner Eng 132:95–109
6. Miriyala SS, Mitra K (2019) Multi-objective optimization of iron ore induration process using optimal neural networks. Mater Manuf Process. https://doi.org/10.1080/10426914.2019.1643476
7. Miriyala SS, Subramanian VR, Mitra K (2018) TRANSFORM-ANN for online optimization of complex industrial processes: casting process as case study. Eur J Oper Res 264(1):294–309
8. Mitra K, Ghivari M (2006) Modeling of an industrial wet grinding operation using data-driven techniques. Comput Chem Eng 30(3):508–520
9. Chelgani SC, Shahbazi B, Rezai B (2010) Estimation of froth flotation recovery and collision probability based on operational parameters using an artificial neural network. Int J Miner Metall Mater 17(5):526–534
10. Yang C et al (2011) Soft sensor of key index for flotation process based on sparse multiple kernels least squares support vector machines. Chin J Nonferr Met 21(12):3149–3154
11. Kaijun, Z., et al., *Flotation recovery prediction based on froth features and LS-SVM [J].* Chinese Journal of Scientific Instrument, 2009. **6**
12. Chelgani SC, Shahbazi B, Hadavandi E (2018) Support vector regression modeling of coal flotation based on variable importance measurements by mutual information method. Measurement 114:102–108
13. Shahbazi B, Chelgani SC, Matin S (2017) Prediction of froth flotation responses based on various conditioning parameters by random forest method. Colloids Surf A 529:936–941
14. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436
15. Miriyala SS, Mitra K (2020) Deep learning based system identification of industrial integrated grinding circuits. Powder Technol 360:921–936
16. Filippov L, Severov V, Filippova I (2014) An overview of the beneficiation of iron ores via reverse cationic flotation. Int J Miner Process 127:62–69
17. Elman JL (1990) Finding structure in time. Cogn Sci 14(2):179–211
18. Gers FA, Schmidhuber J, Cummins F (1999) Learning to forget: continual prediction with LSTM. pp 850–855
19. Cybenko G (1989) Approximations by superpositions of a sigmoidal function. Math Control Signals Syst 2:183–192
20. Pascanu R et al (2013) How to construct deep recurrent neural networks. arXiv preprint arXiv:1312.6026
21. Kingma DP, Ba J (2014) Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980
22. Hua J et al (2004) Optimal number of features as a function of sample size for various classification rules. Bioinformatics 21(8):1509–1515
23. Liaw A, Wiener M (2002) Classification and regression by random forest. R News 2(3):18–22