Fault Detection in a Fluid Catalytic Cracking Process using Bayesian Recurrent Neural Network

Gustavo R. Taira*. Song W. Park*. Antônio C. Zanin*. Carlos R. Porfirio.**

**Department of Chemical Engineering, Polytechnic School, University of Sao Paulo, Sao Paulo, Brazil (e-mails: gustavo.taira@usp.br; sonwpark@usp.br; antczanin@gmail.com)*

***Petrobras S.A., Santos, Brazil (e-mail: carlos.porfirio@petrobras.com.br)*

Abstract: Process safety is still an issue in modern chemical industries. Accidents in chemical processes are still frequent and cause great losses for chemical industries. In this context, there is a demand for the development of intelligent fault detection and diagnosis (FDD) methods that can help operators manage chemical process faults. Since a large amount of process data has become available for monitoring systems as a result of the huge deployment of computer systems and information technologies in chemical industries, the study of data-based FDD methods has become the focus of this research area. Therefore, this work proposes to investigate the performance of a promising Bayesian recurrent neural network-based method in the detection of faults in a real chemical process. The case study is related to the detection of a specific type of fault in a real fluid catalytic cracking process. The method presented satisfactory performance during testing experiments, with a good accuracy detection and a very small number of false-negative cases.

Keywords: process safety, fault detection, fluid catalytic cracking process, machine learning, Bayesian recurrent neural network

1. INTRODUCTION

In the current development process of modern chemical industries, there is a demand for technologies that can improve the quality, efficiency, and safety of chemical processes. Specifically, regarding safety problems, accidents in chemical processes are still very common and cause great losses for chemical industries (Shu *et al.*, 2016). One of the explanations for the still frequent occurrence of accidents is that chemical process monitoring systems are still heavily dependent on human operators for the detection and diagnosis of faults (Wu and Zhao, 2018). This strong dependence on human operators makes monitoring systems vulnerable to operator errors and consequently creates risks to the safety of chemical processes. Therefore, there is a demand to develop intelligent fault detection and diagnosis (FDD) systems that can help operators manage chemical process faults.

Over the last decades, several FDD methods have been proposed in the literature to contribute to the development of these systems. In general, these methods can be categorized into three classes: model-based methods, knowledge-based methods, and data-based methods (Chiang, Braatz and Russell, 2001). However, with the increasing deployment of computer systems and information technologies in chemical industries, a large amount of process measurement data has become available for monitoring systems. Therefore, the study of data-based FDD methods has become the focus of this research area (Yin *et al.*, 2012; Ge, Song and Gao, 2013). Specifically for the detection of faults in chemical processes, among data-based methods, methods based on principal component analysis (PCA) and partial least square (PLS) have been proposed and used in industry for years due to their simplicity of implementation (Qin, 2012). However, these methods are limited to specific cases in which the process data has a stationary behavior and the process variables are linearly correlated. Chemical process data usually have more complex characteristics such as nonlinearity, time-varying, and multimodal behaviors (Ge, Song and Gao, 2013). Therefore, several methods capable of modeling these characteristics of chemical process data have been proposed in the literature, such as dynamic and kernel variations of PCA and PLS methods, and artificial neural network methods (ANN) (Cheng *et al.*, 2019).

Recently, the Bayesian Recurrent Neural Network (BRNN) based method proposed by Sun *et al.*, 2020 was found to be a promising option for detecting faults in chemical processes. Sun *et al.*, 2020 show that its BRNN-based method deals well with non-linear and dynamical characteristics found in chemical process data and provides a probabilistic structure that allows more sensitive and robust detections. Sun *et al.*, 2020 evaluated the method using a dataset generated by a chemical process simulation and a real chemical process dataset, and in both cases, it presented a good performance. Regarding studies about the application of this method to real chemical processes, the case study presented by Sun *et al.*, 2020 is the only one found in the literature. Therefore, this work proposes to investigate the performance of this BRNN-based method in the detection of faults in another real chemical process. This work's case study is about the detection of a specific type of fault that occurs in a real fluid catalytic cracking process.

2. METHODOLOGY

2.1 Recurrent Neural Network

A recurrent neural network (RNN) consists of a specific type of neural network for processing sequential data (Goodfellow, Bengio and Courville, 2016). This type of neural network is characterized by using a feedback loop in its structure (Fig. 1) that allows a 'memory' of previous inputs to persist in the network's internal state and thereby influence the network output (Graves, 2012). Thus, when analyzing sequential data, in addition to being able to model the nonlinearity of the data, such as the traditional neural networks, the RNNs are also able to model the dependence between different sequential data points.

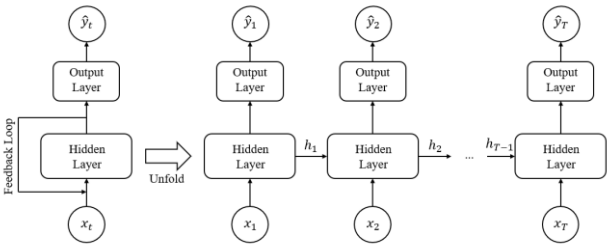

Figure 1 – RNN feedback loop

The structure of a standard RNN hidden layer (Olah, 2015) is illustrated in Fig. 2 and its operation can be described by the following mathematical formulation. Given an input sequence $x = (x_1, \dots, x_T)$, a standard RNN computes the hidden vector sequence $h = (h_1, \dots, h_T)$ and the output vector sequence $\hat{y} = (\hat{y}_1, \dots, \hat{y}_T)$ by iterating the following equations from $t = 1$ to $t = T$:

$$h_t = tanh(W_h \cdot [h_{t-1}, x_t] + b_h) \quad (1)$$

$$\hat{y}_t = W_{hy} \cdot h_t + b_y \quad (2)$$

where $\cdot$ denotes the dot product, $[h_{t-1}, x_t]$ denotes the concatenation of the last hidden state h_{t-1} with the current input x_t, W_h and W_{hy} are weight matrices, b_h and b_y are bias vectors, and $tanh(\cdot)$ is a hyperbolic tangent activation function.

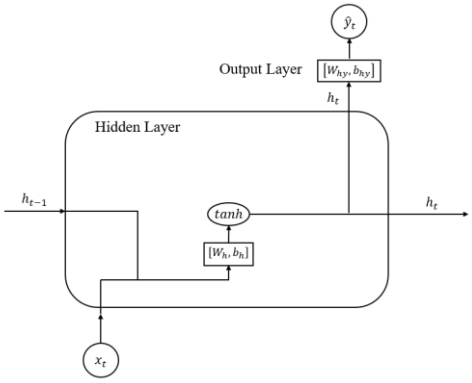

Figure 2 – Standard RNN hidden layer

For training this type of neural network, the backpropagation through time (BPTT) algorithm (Williams and Zipser, 1995) is normally used, which is a specific variation of the backpropagation algorithm (Rumelhart, Hinton and Williams, 1985) for RNNs.

In theory, standard RNNs should be able to learn long-term temporal dependencies between sequential data points, however, this does not happen in practice due to the gradient vanishing problem. This problem causes the gradient values calculated during the training process to become extremely small, affecting the network learning (Hochreiter and Schmidhuber, 1997). In this context, variations of RNNs, such as Long-Short Term Memory networks (Hochreiter and Schmidhuber, 1997), have been widely used in practice since they use robust architectures that avoid the gradient vanishing problem during the training process (Chung *et al.*, 2014).

2.2 Long Short-Term Memory Network

A Long Short-Term Memory (LSTM) network consists of a specific type of recurrent neural network that uses memory gates in its architecture to mitigate the gradient vanishing problem during its training (Hochreiter and Schmidhuber, 1997). Unlike the standard RNN (Fig. 2) where the hidden layer only contains one data processing layer (represented by the box $[W_h,\ b_h]$), an LSTM hidden layer, as illustrated in Fig. 3, contains four processing layers (represented by boxes $[W_f,\ b_f], [W_i,\ b_i], [W_g,\ b_g], [W_o,\ b_o]$), each with a specific function. These processing layers are used to protect and control the state cell (C_t) of the LSTM network, represented by the upper horizontal line in the hidden layer illustrated in Fig. 3, which has the function of transmitting information obtained from already analyzed data points to the analysis of new data points. The processing layer $[W_f,\ b_f]$ is used as a forget gate that decides which information should be discarded from the state cell, $[W_i,\ b_i]$ is used as an input gate that controls what information should be stored in the state cell, $[W_g,\ b_g]$ is used in conjunction with the input gate providing possible new information $\tilde{C}_t$ for the state cell, and $[W_o,\ b_o]$ is used as an output gate that decides what information should be made available for the output of the LSTM network (Olah, 2015).

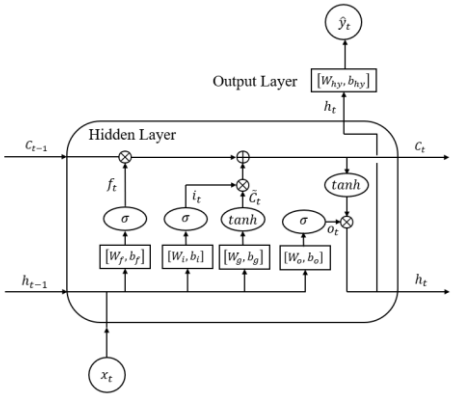

Figure 3 – LSTM hidden layer

Given an input x_t, the operation of a basic LSTM network to calculate its respective output $\hat{y}_t$ is described by the following equations:

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \quad (3)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (4)$$

$$\tilde{C}_t = tanh\left(W_g \cdot [h_{t-1}, x_t] + b_g\right) \quad (5)$$

$$C_t = f_t \otimes C_{t-1} + i_t \otimes \tilde{C}_t \quad (6)$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (7)$$

$$h_t = o_t \otimes tanh(C_t) \quad (8)$$

$$\hat{y}_t = W_{hy} \cdot h_t + b_y \quad (9)$$

σ denotes a sigmoid activation function and $\otimes$ denotes a Hadamard product.

2.3 Bayesian Recurrent Neural Network

A Bayesian recurrent neural network (BRNN) is a probabilistic interpretation of an RNN based on a Bayesian modeling concept. In this modeling approach, given a training dataset with $X = (x_1, \dots, x_T)$ as its input data and $Y = (y_1, \dots, y_T)$ as its respective target output data, a parameterized model of an RNN, denoted by $Y = F^W(X)$, is trained to determine the most likely network's parameters W to have generated the data. To do this, a Bayesian learning process is used to train the model.

In a Bayesian learning process, a prior distribution $p(W)$ is initially defined to represent the prior beliefs about which parameter values are likely to have generated the training data before any analysis of the data. As soon as the training data is analyzed by the model, this prior distribution $p(W)$ is then updated to determine the distribution that describes the most likely parameters' values according to the information obtained from the analyzed data. For this update to be carried out, a likelihood distribution $p(Y|X,W)$ is defined, which is used to describe the information of W derived from the training data. In the case of a regression problem this distribution can be represented by a Gaussian model:

$$p(Y|X,W) = \prod_{t=1}^{T} p(y_t|x_t,W) \quad (10)$$

$$p(y|x,W) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{(f^W(x)-y)^2}{2\sigma^2}\right) \quad (11)$$

Thus, given the likelihood distribution $p(Y|X,W)$ and the prior distribution $p(W)$, the posterior distribution $p(W|X,Y)$ can be calculated using the Bayes' theorem:

$$p(W|X,Y) = \frac{p(Y|X,W)p(W)}{p(Y|X)} \quad (12)$$

$p(Y|X)$ denotes a normalization term.

This posterior distribution $p(W|X,Y)$ describes which are the most likely values for the parameters of the model to have generated the data, according to the combination of prior knowledge about the parameters and information about the parameters provided by the observed data. By the end of the posterior distribution computation, the Bayesian learning process is completed. The computed posterior distribution then can be used by the model to predict the most likely output values $\hat{y}_t$ for new input values x_t through the following integration:

$$p(\hat{y}_t|x_t,X,Y) = \int p(\hat{y}_t|x_t,W)p(W|X,Y)dW \quad (13)$$

This prediction operation is called Bayesian inference (Gal, 2016). Unlike a traditional RNN model, where parameters and predictions assume punctual numerical values, in a BRNN, probability distributions are inferred over its parameters and predictions, thus providing advantages such as the representation of their uncertainties and robustness against overfitting (Gal, 2016). However, due to the fact that calculating the posterior distribution $p(W|X,Y)$ is not analytically possible for models that contain many parameters, implementing a BRNN is not a simple task. A key component in the posterior distribution computation is the normalization term $p(Y|X)$, which is calculated by the following integration:

$$p(Y|X) = \int p(Y|X,W)p(W)dW \quad (14)$$

This term is normally a limiting factor for the calculation of $p(W|X,Y)$, as it requires the marginalization of the likelihood distribution over all possible values for W, which can only be performed analytically for simple models with a limited number of parameters. Thus, the Bayesian inference in models such as BRNNs usually cannot be done analytically, thus turning necessary in these cases to use Bayesian inference approximation methods (Gal, 2016). Since the introduction of the Bayesian neural network concept in the 90s (MacKay, 1992; Neal, 1995), several methods for approximating Bayesian inference in neural networks have been proposed. Most of these methods are based on variations of traditional Bayesian inference approximation methods such as Markov Chain Monte Carlo methods, method of Laplace, and variational inference methods. A review of Bayesian inference approximation methods specific to Bayesian neural networks is presented by Jospin *et al.*, 2020. For this work, the variational inference method called Monte Carlo (MC) Dropout (Gal and Ghahramani, 2016b) is adopted, since it allows to approximately perform Bayesian inference in BRNNs with only simple adaptations of traditional training techniques for RNNs (Gal and Ghahramani, 2016a). This method is based on the interpretation of the dropout regularization technique (Srivastava *et al.*, 2014) as a variational inference method that uses a Bernoulli distribution as a posterior variational distribution.

2.4 BRNN-based Fault Detection Method

As presented by Sun *et al.*, 2020, for a BRNN model to be used as a fault detection model, it must be trained offline using data from normal process operation to learn how to characterize the normal process operating conditions. Specifically, as illustrated in Fig. 4, the BRNN model must learn to receive as

input process monitoring data collected at one instant t and provide in response a predictive distribution of values for monitoring data at the next instant $t+1$.

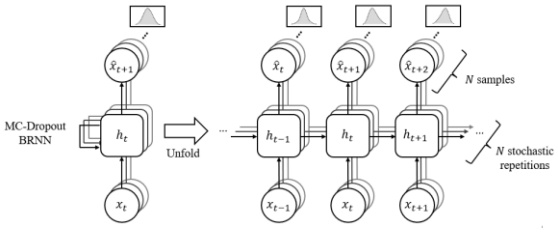

Figure 4 - BRNN model for fault detection

Thus, after training, when receiving a new monitoring data x_t as input, the model is able to provide a predictive distribution of expected values for the process monitoring data in the next instant $\hat{x}_{t+1}$ in the case of normal operation condition. Upon being collected, the actual monitoring data x_{t+1} is then compared to the predictive distribution $\hat{x}_{t+1}$ provided by the model. If a significant deviation of the data x_{t+1} from the predictive distribution $\hat{x}_{t+1}$ is detected, then the data x_{t+1} is classified as a data that describes the occurrence of a fault in the process. Otherwise, the data is classified as normal operation data. The complete scheme for fault detection in a chemical process using a BRNN model is described in Fig.5.

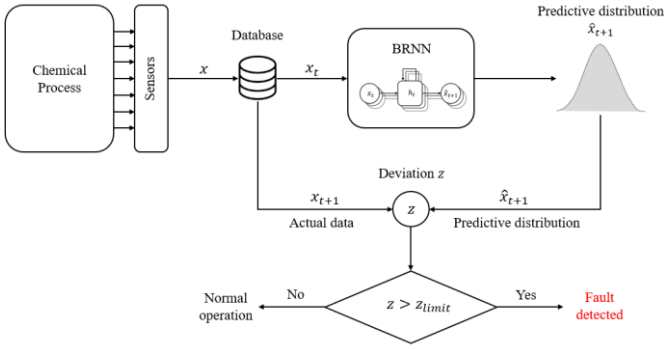

Figure 5 – BRNN-based fault detection method

3. CASE STUDY

To investigate the performance of the BRNN-based method in the detection of real chemical processes faults, it is proposed a case study related to the detection of a specific type of fault that occurs in a real fluid catalytic cracking (FCC) process. The FCC process is one of the main chemical processes in an oil refinery as it cracks heavier oil fractions into light products with higher added value, such as gasoline and liquefied petroleum gas (LPG). The FCC process that is the focus of this case study is characterized by containing a large number of sensors that collect process operation data and, consequently, by having available a large volume of historical operation data related to the occurrence of different types of faults in the process. Specifically for this case study, it is proposed to investigate the performance of the BRNN-based method in the detection of an FCC reactor fault caused by a sticking problem in its catalyst level control valve. According to the engineers of this process, the detection of this type of fault is still carried out manually by the operators, and the development of an intelligent model that helps them detect this failure would be very useful.

4. EXPERIMENT

4.1 FCC Process Data

Three historical operation data windows of the FCC process reactor were provided for the development of the case study, a normal operation data window, and two data windows that contain reactor operation data under the occurrence of faults caused by the sticking problem in the catalyst level control valve. The normal operation data window was used as the training set. The other two data windows were used as test sets and were denoted as test set 1 and test set 2, respectively. The training set contained 15 days of reactor monitoring data (20161 samples), the test set 1 contained 11 days of reactor monitoring data (14401 samples), and the test set 2 contained 10 days reactor of monitoring data (12961 samples). The samples were collected every 1 minute. A list of the 16 process variables that were available in the datasets is displayed in Table 1.

Table 1 – Available FCC process variables

Variable	Description
x^1	Reactor upper vessel level
x^2	Reactor upper vessel level - set point
x^3	Reactor upper vessel level - manipulated variable
x^4	Feedback output
x^5	Pressure difference at the regenerator valve
x^6	Pressure difference at the regenerator valve - set point
x^7	Pressure difference at the regenerator valve - manipulated variable
x^8	Heavy gas oil feed flow
x^9	Nafta feed flow
x^{10}	Regenerator pressure
x^{11}	Reactor pressure
x^{12}	Reaction temperature
x^{13}	Reaction temperature - set point
x^{14}	Reaction temperature - manipulated variable
x^{15}	Feed temperature
x^{16}	Air flow in the regenerator

4.2 Data Selection and Preprocessing

In order to avoid high-dimensioned problems, only the most relevant variables to detect the fault were selected to train the model. According to the engineers of the FCC process focus of the case study, the process variable that describes the level of the reactor's upper vessel (x^1) is the variable that suffers the most noticeable disturbances during the occurrence of a sticking problem in the catalyst level control valve. Thus, this variable was the first one to be selected. In addition, the variables with the highest correlation to the variable x^1 and consequently can help in the detection of valve failures were searched. By calculating the correlation matrix of the variables, it was found that x^3 and x^5 are the variables most correlated to the variable x^1. The other variables had a very low or almost null correlation to the variable x^1. Thus, only the variables x^1, x^3, and x^5 were selected for training the model. Regarding the pre-processing of data, the only action needed was its normalization before training.

4.3 BRNN Model

To carry out the experiment, due to the advantage of LSTM networks against the gradient vanishing problem, a Bayesian LSTM recurrent neural network model was chosen. The model was structured with 3 input units, a hidden LSTM layer with memory gates containing 32 processing units, and an output layer with 1 processing unit. The model was structured in a way it can receive as inputs the values of variables x_t^1, x_t^3 and x_t^5 at timestep t and return a predictive distribution $\hat{x}_{t+1}^1$ of probable values of variable x^1 at the time $t+1$. This predictive distribution is then compared to the actual value of x_{t+1}^1 to decide if a fault is occurring at instant $t+1$.

4.4 Model Training

As described above, the proposed model is trained using only the reactor's normal operation data window so it learns to predict the values of the variable x^1 in the normal operating condition of the reactor. For the training process, the Bayesian inference approximation method MC-Dropout was used. This method, in the case of BRNNs, basically consists of the application of the BPTT algorithm to data sequence batches with a variational dropout mask sampled for each sequence. For training the proposed model, the dropout rate was set to 0.3 and the optimizer Adam (Kingma and Ba, 2015) was utilized with a learning rate equal to 0.001. The training was carried out for 20 epochs. The model was implemented using Python programming language and the TensorFlow library (TensorFlow, 2021).

4.5 Model Test

After training, following the fault detection method presented in section 2.4, the model was submitted to the two test sets and performed the classification of each data point in these windows as normal operation data or faulty operation data. To assess the performance of the trained model in the detection of the reactor's faults, the number of true positives (TP), false positives (FP), false negatives (FN) e true negatives (TN) were calculated. With these four values computed, the model accuracy (Acc), its true-positive rate (TPR) and its true-negative rate (TNR) were then calculated using the following equations:

$$Acc = \frac{TP + TN}{TP + FP + TN + FN} \tag{15}$$

$$TPR = \frac{TP}{TP + FN} \tag{16}$$

$$TNR = \frac{TN}{FP + TN} \tag{17}$$

And, since there is an imbalance between the amount of normal operation data and the amount of faulty operation data, the balanced accuracy ($b.Acc$) (Brodersen *et al.*, 2010) was also calculated:

$$b.Acc = \frac{\frac{TP}{TP + FN} + \frac{TN}{FP + TN}}{2} \tag{18}$$

5. RESULTS

Table 2 presents a summary of the BRNN-based method's detection performance for each of the test sets. Fig. 6 and Fig. 7 illustrate the performance of the method in the detection of faults in the test set 1 and test set 2, respectively.

Table 2 – Method's performance summary

Performance Metric (%)	Test Set 1	Test Set 2
TP	0.94	1.56
FP	0.49	0.62
FN	0.01	0.01
TN	98.56	97.82
TPR	98.95	99.51
TNR	99.51	99.37
Acc	99.50	99.38
b.Acc	99.23	99.44

As can be seen in Table 2, for test dataset 1 the model had an accuracy of 99.50%, a balanced accuracy of 99.23%, a true-positive rate of 98.95%, a true-negative rate of 99.51% and only 0.01% of false-negative cases. For test data set 2, the model presented an accuracy of 99.38%, a balanced accuracy of 99.44%, a true-positive rate of 99.51%, a true-negative rate of 99.37% and only 0.01% of false-negative cases.

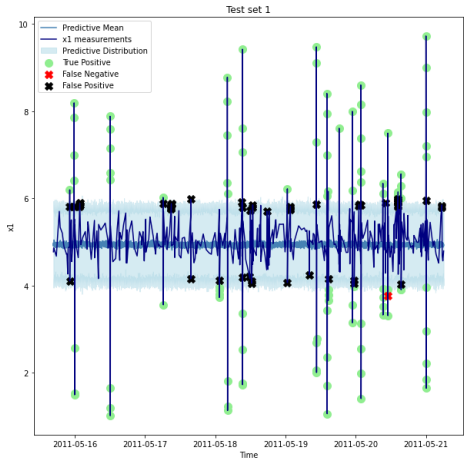

Figure 6 - BRNN model performance illustration in detecting faults in test set 1

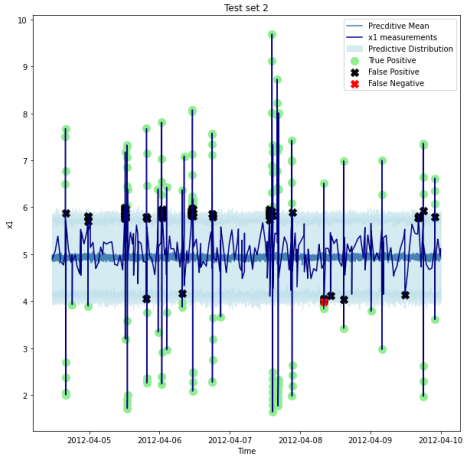

Figure 7 - BRNN model performance illustration in detecting faults in test set 2

6. CONCLUSION

An application of a Bayesian recurrent neural network-based method is proposed for the detection of faults in a real chemical process. The case study is related to the detection of a specific type of fault in a real FCC process. In general, the method presented satisfactory accuracy during testing experiments, with a very small number of false negative cases. Therefore, the method proves to be a promising method for detecting faults in real chemical processes.

ACKNOWLEDGMENT

The authors thank the Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) for the financial support (Processo 19/08280-9).

REFERENCES

Brodersen, K. H. *et al.* (2010) 'The balanced accuracy and its posterior distribution', *Proceedings - International Conference on Pattern Recognition*, pp. 3121–3124. doi: 10.1109/ICPR.2010.764.

Cheng, F. *et al.* (2019) 'A robust air balancing method for dedicated outdoor air system', *Energy and Buildings*, 202. doi: 10.1016/j.enbuild.2019.109380.

Chiang, L. H., Braatz, R. D. and Russell, E. L. (2001) *Fault detection and diagnosis in industrial systems*. Springer Science & Business Media.

Chung, J. *et al.* (2014) 'Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling', pp. 1–9. Available at: http://arxiv.org/abs/1412.3555.

Gal, Y. (2016) *Uncertainty in Deep Learning*. University of Cambridge.

Gal, Y. and Ghahramani, Z. (2016a) 'A theoretically grounded application of dropout in recurrent neural networks', in *Advances in Neural Information Processing Systems*. Neural information processing systems foundation, pp. 1027–1035.

Gal, Y. and Ghahramani, Z. (2016b) 'Dropout as a Bayesian approximation: Representing model uncertainty in deep learning', in *33rd International Conference on Machine Learning, ICML 2016*. International Machine Learning Society (IMLS), pp. 1651–1660.

Ge, Z., Song, Z. and Gao, F. (2013) 'Review of recent research on data-based process monitoring', *Industrial and Engineering Chemistry Research*, 52(10), pp. 3543–3562. doi: 10.1021/ie302069q.

Goodfellow, I., Bengio, Y. and Courville, A. (2016) *Deep learning*. MIT Press.

Graves, A. (2012) *Supervised sequence labelling with recurrent neural networks*. Springer.

Hochreiter, S. and Schmidhuber, J. (1997) 'Long Short-Term Memory', *Neural Computation*, 1735–1780(9), pp. 1735–1780.

Jospin, L. V. *et al.* (2020) 'Hands-on Bayesian Neural Networks - a Tutorial for Deep Learning Users', 1(1), pp. 1–35. Available at: http://arxiv.org/abs/2007.06823.

Kingma, D. P. and Ba, J. L. (2015) 'Adam: A method for stochastic optimization', *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–15.

MacKay, D. J. C. (1992) 'A Practical Bayesian Framework for Backpropagation Networks', *Neural Computation*, 4(3), pp. 448–472. doi: 10.1162/neco.1992.4.3.448.

Neal, R. M. (1995) *Bayesian Learning for Neural Networks*. University of Toronto. doi: 10.2307/2965731.

Olah, C. (2015) *Understanding LSTM networks*. Available at: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Qin, S. J. (2012) 'Survey on data-driven industrial process monitoring and diagnosis', *Annual Reviews in Control*, 36(2), pp. 220–234. doi: 10.1016/j.arcontrol.2012.09.004.

Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1985) *Learning Internal Representations By Error Propagation*, *Explorations in the Micro-Structure of Cognition Vol. 1 : Foundations*.

Shu, Y. *et al.* (2016) 'Abnormal situation management: Challenges and opportunities in the big data era', *Computers and Chemical Engineering*, 91, pp. 104–113. doi: 10.1016/j.compchemeng.2016.04.011.

Srivastava, N. *et al.* (2014) 'Dropout: A Simple Way to Prevent Neural Networks from Overfitting', *Journal of Machine Learning Research*, pp. 1929–1958. doi: 10.1016/0370-2693(93)90272-J.

Sun, W. *et al.* (2020) 'Fault detection and identification using Bayesian recurrent neural networks', *Computers and Chemical Engineering*, 141, p. 106991. doi: 10.1016/j.compchemeng.2020.106991.

TensorFlow (2021) *TensorFlow*. Available at: https://www.tensorflow.org/.

Williams, R. J. and Zipser, D. (1995) 'Gradient-based learning algorithms for recurrent', *Backpropagation: Theory, architectures, and applications*, 433.

Wu, H. and Zhao, J. (2018) 'Deep convolutional neural network model based chemical process fault diagnosis', *Computers and Chemical Engineering*, 115, pp. 185–197. doi: 10.1016/j.compchemeng.2018.04.009.

Yin, S. *et al.* (2012) 'A comparison study of basic data-driven fault diagnosis and process monitoring methods on the benchmark Tennessee Eastman process', *Journal of Process Control*, 22(9), pp. 1567–1581. doi: 10.1016/j.jprocont.2012.06.009.