Fault Detection and Diagnosis in a Chemical Process using Long Short-Term Memory Recurrent Neural Network

Gilberto M. Xavier*
*Petrobras Research & Development Center
Rio de Janeiro, Brazil
gilberto.xavier@petrobras.com.br

José Manoel de Seixas†
*†Signal Processing Laboratory, COPPE/Poli
Federal University of Rio de Janeiro
Rio de Janeiro, Brazil
seixas@lps.ufrj.br

***Abstract*—Long short-term memory recurrent neural networks have been proved to be especially useful for learning sequences comprising longer-term patterns of unknown length as they are able to preserve long-term memory. The learning of higher level temporal features could be achieved by stacking recurrent hidden layers for faster learning with sparser representations. In this work, we propose a novel approach to data driven fault detection and diagnosis of a chemical process. The method employs a state-of-the-art deep-earning technique, viz. the long short-term memory recurrent neural network. An application of the proposed approach is performed with realistic simulated data from a chemical process literature benchmark. Results indicate an excellent performance when compared to already published linear and nonlinear fault detection and diagnosis methods.**

***Index Terms*—deep networks, recurrent neural network, long short-term memory, fault detection and diagnosis, Tennessee Eastman chemical process**

I. Introduction

Failures of plant equipment and instrumentation produce a negative impact on the safety of any process plant. In addition, the consequences of a gross accident such as an explosion due faulty design or operation are even more serious [1].

For example, on 2010 a safety equipment failure led to the catastrophic Macondo incident resulting in the deaths of 11 workers, the sinking of the Deepwater Horizon rig, and massive marine and coastal damage, making it one of the largest environmental disasters in US history [2].

Modern oil rigs, petrochemical and chemical plants share many similarities as they are characterized by:

- complex processes and pieces of equipment;
- high throughput;
- long sequential process trains with considerable recycle;
- complex controls and instrumentation that compensate for and conceal faults, and;
- serious consequences for a catastrophic accident.

Demands for higher safety, greater output of consistent products, reliability and quality using less energy are being pushed up in the chemical, petrochemical and oil & gas industries, as well as to satisfy government stringent environmental standards [3].

Therefore, to meet these higher demands, the complexity and automation of modern process plants have significantly increased [4], [5]. For example, in a modern petrochemical plant for xylenes and benzene production, there is a great number of sensors and actuators embedded that are paired to form several control loops [6].

These standard control loops are designed to maintain satisfactory operations by compensating for the effects of disturbances and changes occurring in the process, but there are changes in the process that these control loops cannot handle adequately. These changes are called faults and they are defined as unpermitted deviations of at least one characteristic property or variable of the process [4].

The trend of increasing complexity and automation of process plants allied with the current technological advances in communication networks and information technology lead to a huge availability of operation data relating to the processs conditions and status. As a result, any system of fault detection and diagnosis that extracts meaningful information from these data deluge and permits the use of less expensive equipment, increases plant availability, reduces maintenance costs and increases safety merits serious attention.

Fault detection and diagnosis (FDD) deals with the problem of how to detect the occurrence of a fault as early as possible and how to identify which fault occurred as accurately as possible [4]. However, most existing methods for FDD in chemical and petrochemical processes developed over the past decades [1], [4], [7] are unable to fully and effectively consume the huge amount of the data collected by the modern distributed control systems. In consequence and as cited by [8], *large volumes of data with very little information* is a quite common phenomenon in today's industrial automation.

This phenomenon opened up an interesting opportunity window to investigate the use of deep-learning data hungry techniques [9] in FDD applications. Keeping up with this trend, this work proposes a new paradigm for the use of a state-of-the-art deep-learning algorithm like the long short-

J. M. S. thanks FAPERJ, CAPES and CNPq (Brazil) for their support during this work.

term memory recurrent neural network [10] in a chemical process FDD problem.

In a nutshell, the novel approach presented here differs from the already published [11]–[14] in three key aspects:

- it allows, besides the detection, also the fault diagnosis;
- it doesn't need any tunable detection threshold or a sequence-to-sequence prediction, and;
- it solves the FDD problem using an input with much lower dimensionality when compared to [14].

The remainder of this paper is organized as follows. First, some preliminaries on recurrent neural networks are given. Thereafter, Section III describes the Tennessee Eastman benchmark process used as the source of the sequential data set. In Section IV, the experiment for the evaluation of the recurrent neural network used to detect and diagnosis the faults is described and the results obtained are compared with the literature. In the last section, conclusions and possible future studies are summarized.

II. Recurrent Neural Networks

Recurrent neural networks or RNNs [15] represent a family of neural networks for processing sequential data [16, p.373]. This family of neural networks is widely applied in the scientific literature for deep learning in different pattern recognition tasks and frequently outperforming classical approaches when the task involves large and complex sequential data sets [17].

Recurrency or feedback means that information is no longer transmitted only in one direction but it is also transmitted backwards in contrast with the classical feed-forward neural network topology [18]. The feedback creates an internal state or memory which allows the RNNs to exhibit dynamic temporal behaviour and thus be able to learn contextual information within the sequential data [19].

A. Vanilla RNN

A finite size recurrent neural network is able to compute any function computable by a Turing machine [16, p. 379]. Given an input sequence $\mathbf{x} = (x_1, ..., x_T)$, a standard RNN computes the hidden vector sequence $\mathbf{h} = (h_1, ..., h_T)$ and the output vector sequence $\hat{\mathbf{y}} = (\hat{y}_1, ..., \hat{y}_T)$ by iterating the following equations from $t = 1$ to T:

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

$$\hat{y}_t = W_{hy} \cdot h_t + b_y \tag{2}$$

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

Alternatively, as in a *many to one* classification setting [20, p. 13], the output is computed by:

$$\hat{y} = W_{hy} \cdot h_T + b_y \tag{3}$$

From a theoretical point of view, this vanilla RNN is able to learn *long-term* dependencies. Unfortunately, empirical evidence [10, Table 2, p. 1754] shows that RNN with this simple repeating module structure don't seem to be able to learn *long-term* dependencies when the number of steps T is greater than 10. This limitation is basically due to the vanishing gradient problem for which the solution is given by a more complex RNN family called Long Short-Term Memory [10].

B. Long Short-Term Memory (LSTM)

LSTM is a RNN family really capable to learn *long-term* dependencies because it has an anti vanishing gradient built-in mechanism [21]. Despite having the same chain like structure as the vanilla RNN presented earlier, the difference between the two is the structure of their repeating module A. A basic LSTM repeating module A has four neural network layers represented by yellow boxes in Fig. 1.

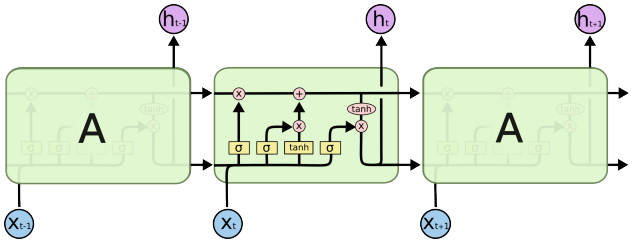

Fig. 1. A basic one layer LSTM repeating module structure (reproduced with permission from [22]).

Three of these layers have the logistic function as the activation function and are called gates. The gates are used by the LSTM to protect and control its repeating module or cell state which is represented by the horizontal line running through the top of the structure. The LSTM decides what information it will throw away from the cell state using the forget gate output f_t, which has the following forward propagation equation:

$$f_t = \sigma(W_h \cdot [h_{t-1}, x_t] + b_h) \tag{4}$$

where $\sigma(\cdot)$ is the the logistic function.

The same approach is used by the LSTM to store new information in its cell state and this is controlled by the output i_t of the input gate:

$$i_t = \sigma(W_h \cdot [h_{t-1}, x_t] + b_h) \tag{5}$$

The new information is given by the LSTM internal cell state layer output $\widetilde{C}_t$:

$$\widetilde{C}_t = tanh(W_h \cdot [h_{t-1}, x_t] + b_h) \tag{6}$$

The LSTM cell state is then updated by the following equation:

$$C_t = f_t \odot C_{t-1} + i_t \odot \widetilde{C}_t \tag{7}$$

where $\odot$ is the Hadamard product.

Finally, the LSTM decides what information will belong to its output using the output gate layer o_t:

$$o_t = \sigma(W_h \cdot [h_{t-1}, x_t] + b_h) \tag{8}$$

The LSTM hidden state h_t is then calculated by the following equation:

$$h_t = o_t \odot tanh(C_t) \tag{9}$$

and its output is computed by Eqs. 2 or 3.

The basic LSTM structure presented here [23] is one of several different LSTM RNN variants proposed since LSTM was designed by [10]. For more information on RNNs and LSTM, we refer the reader to [24].

III. Tennessee Eastman process

Originally created by Downs and Vogel as a process control challenge problem in the earlier 90's [25], [26], the Tennessee Eastman process (TE process) is a realistic simulation of an open loop industrial plant based on a real chemical process as well a benchmark process that has been widely used in fault detection and diagnosis studies [27]–[32].

The TE process is based on an actual chemical process simulation where the components, kinetics, and operating conditions have been modified due confidentiality issues. The entire process consists of five major units: a reactor, a product condenser, a recycle compressor, a vapor-liquid separator, and a product stripper performing two simultaneous gas-liquid exothermic reactions of the following form:

$$A(g) + C(g) + D(g) \rightarrow G(l), \text{ Product 1} \tag{10}$$

$$A(g) + C(g) + E(g) \rightarrow H(l), \text{ Product 2} \tag{11}$$

Besides the reagents A, C, D and E, an inert gas B is also introduced into the reactor feed stream and further two additional byproduct reactions occur:

$$A(g) + G(g) \rightarrow F(l), \text{ Byproduct} \tag{12}$$

$$3D(g) \rightarrow 2F(l), \text{ Byproduct} \tag{13}$$

The reactions are irreversible, exothermic, and approximately first-order with respect to the reactant concentrations. The reaction rates are Arrhenius functions of temperature where the reaction for *G* has a higher activation energy than the reaction for *H*, resulting in a higher sensitivity to temperature.

The entire plant-wide process piping and instrumentation diagram is given in Fig. 2. The reactor product stream is cooled through a partial condenser and then fed to a vapor-liquid separator. The vapor exiting the separator is recycled to the reactor feed through a compressor. A portion of the recycle stream is purged to keep the inert and byproduct from accumulating in the process. The condensed components from the separator (stream 10) are pumped to a stripper. Stream 4 is used to strip the remaining reactants from stream 10, which are combined with the recycle stream via stream 5. Products *G* and *H* exiting the bottom of the stripper are sent to a downstream process.

The original TE process was made available by the Eastman Chemical Company, formerly called Tennessee Eastman Company, in open loop operation. Since this process is open loop unstable and chemical processes are, in reality, operated under closed loop, a plant-wide control scheme must be employed when applying the fault detection and diagnosis methods.

A number of studies regarding different plant-wide control structures were investigated for the TE process and here the second control structure listed in [33], [34] was chosen, because it provides the best performance according to [34].

The TE-process model operated under closed loop generates values for the 41 measured variables and the 12 manipulated variables listed in Table I. A total of 21 different process faults (see Table II) could be simulated for performance evaluation of the fault detection and diagnosis methods.

Faults y_1 to y_7 are step changes, while faults y_8 to y_{12} are random changes of process variables; fault y_{13} is slow shift of reaction kinetics; faults y_{14}, y_{15} and y_{21} are related to valve sticking and faults y_{16} to y_{20} are types of unknown faults. Among these faults, some faults are easy to detect as they greatly affect the process and change the relations between process variables. However, there are also faults that are difficult to detect (faults y_3, y_9 and y_{15}) because they are very small and have little influence to the measured and manipulated variables.

For detailed descriptions of the original process, as well as the closed loop augmented one, the reader should consult [4], [7], [25], [26], [35], [36].

TABLE I
Manipulated variables

Variable	Description
x_1	D feed flow (stream 2)
x_2	E feed flow (stream 3)
x_3	A feed flow (stream 1)
x_4	A and C feed flow (stream 4)
x_5	Compressor recycle valve
x_6	Purge valve (stream 9)
x_7	Separator pot liquid flow (stream 10)
x_8	Stripper liquid product flow (stream 11)
x_9	Stripper steam valve
x_{10}	Reactor cooling water flow
x_{11}	Condenser cooling water flow
x_{12}	Agitator speed

IV. Experiment

This section presents the data preparation, the LSTM design and result comparisons of the experiment conducted in order to evaluate the performance of the LSTM network in detecting and diagnosing faults inside a subset of the multivariate TE-process sequential data.

Contrary to previous studies and in order to make the problem similar to a dynamic system identification problem, in this work only the 11 manipulated variables[1] are used for

[1]The agitator speed is neglected because it is actually constant.

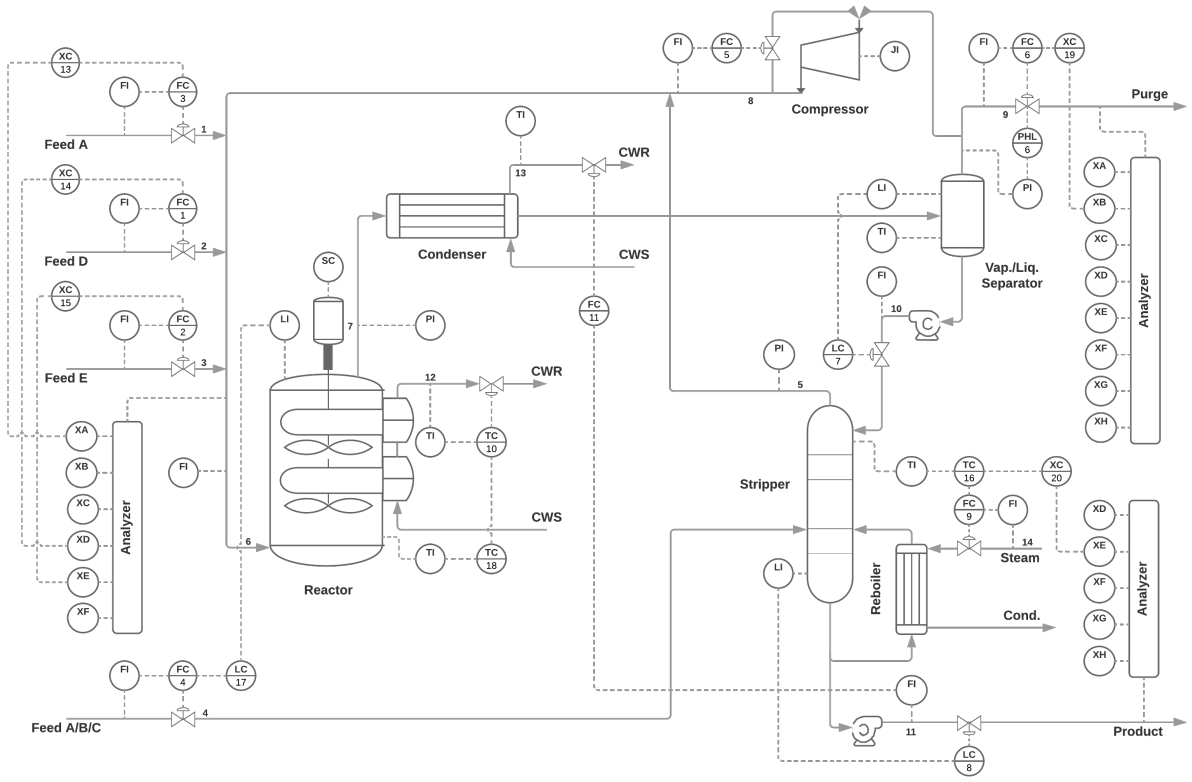

Fig. 2. TE process piping and instrumentation diagram [34, p. 323].

TABLE II
FAULTS OVERVIEW

Fault	Description	Type
y_1	A/C ratio, B composition constant (stream 4)	Step
y_2	B composition, A/C ratio constant (stream 4)	Step
y_3	D feed temperature (stream 2)	Step
y_4	Reactor cooling water supply temperature	Step
y_5	Condenser cooling water supply temperature	Step
y_6	A feed loss (stream 1)	Step
y_7	C header pressure loss (stream 4)	Step
y_8	A, B, C feed composition (stream 4)	Random
y_9	D feed temperature (stream 2)	Random
y_{10}	C feed temperature (stream 4)	Random
y_{11}	Reactor cooling water supply temperature	Random
y_{12}	Condenser cooling water supply temperature	Random
y_{13}	Reaction kinetics	Slow drift
y_{14}	Reactor cooling water valve	Sticking
y_{15}	Condenser cooling water valve	Sticking
y_{16}	Unknown	-
y_{17}	Unknown	-
y_{18}	Unknown	-
y_{19}	Unknown	-
y_{20}	Unknown	-
y_{21}	A, B, C feed valve (stream 4)	Constant position

the FDD task of six step type faults of the TE process, namely y_1, y_2, y_3, y_4, y_5 and y_7[2]. A more comprehensive evaluation comprising all faults is currently being prepared and will be published later.

A. Data preparation

A R package called `tep` [37] is used to generate and visualize the simulated time series data set. This package is actually a wrapper of a modified version of the FORTRAN code for the TE-process closed loop simulation model developed in the earlier 2000's by the Braaz's group [4], [7], [35], [36].

TE-process time series data sets normally employed by the literature to evaluate the performance of the FDD methods, as the ones available at Braaz's group site[3], have a visceral weakness, since the number of normal and fault samples is practically balanced. This characteristic is not true for most real process historic records, as it is expected that the process run in normal condition most of the time making the number of normal and fault samples overly unbalanced in a real situation.

In order to generate a simulated time series data set as similar as possible to a data stream from a real process, we adopt a different approach. This new approach tries to mimic the generation of real process history records and is based on a binary (0 or 1) fault matrix where each column represents one fault and each row is a step. When an entry of the binary fault matrix is equal to 1, it means that the corresponding fault (column) is present at the corresponding step (row), otherwise the fault is not present.

[2]Fault y_6 was not included because it is a remarkable easy one.

[3]http://web.mit.edu/braatzgroup/TE_process.zip

The binary fault matrix is randomly generated with just two constraints: first, there is one and only one fault activated at each step and second, the duration of each fault is limited to a minimum and a maximum value.

The number of columns of this matrix is equal to the number of possible faults, but its row number depends on the span of the process history recorded. In order to obtain enough data to satisfy the data hungry LSTM network, we arbitrarily set the span of the process history recorded to 2^{19} steps[4]. The simulated process history of the 11 manipulated variables corresponding to the first 25000 steps is shown in Fig. 3.

From this time series data we extract a total of 8191 sequences to feed our voracious LSTM network model, where each sequence has a span of 128 simulation samples and is 50% overlapped with its predecessor. The sequences are labeled as normal operation (y_0) or as a specific fault using a *winner-takes-all* voting scheme based on the entries' values of the respective rows of the binary fault matrix. The first 5733 sequences were used as the sequential data training set and the remaining 2458 sequences formed the sequential data test set. The uneven labels balance in each sequential data set could be seen in Table III[5].

TABLE III
LABELS COUNT

Label	Count	
	Training set	Test set
y_0	4709	2030
y_1	172	65
y_2	168	63
y_3	162	75
y_4	117	54
y_5	184	68
y_7	221	103

B. LSTM-based network

Contrary to [11]–[14], our LSTM-based FDD method doesn't employ the LSTM network as a forecasting model neither detects the faults on the base of mean square error between prediction and observation given a detection threshold. Our approach turns the FDD task into a *many to one* classification problem [20, p. 13], where each sequence (*many values*) is classified into a unique fault (*one value*). Thus it doesn't need any tunable detection threshold or a sequence-to-sequence prediction.

The LSTM-based network was implemented based on the work of [38], [39] using Python and the TensorFlow backend [40]. The input layer uses a rectified linear unit (ReLU) activation function and is given by the following equation:

$$h_t^0 = \max\left(0, W_{xh} \cdot x_t + b_x\right) \tag{14}$$

where W_{xh} is a weight matrix and b_x is a bias vector. The LSTM cell structure used is given by [41]:

$$\begin{pmatrix} i_t \\ f_t \\ o_t \\ \widetilde{C}_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \left\{ W^l \cdot \left[h_{t-1}^l, h_t^{l-1} \right] + b^l \right\} \tag{15a}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \widetilde{C}_t \tag{15b}$$

$$h_t^l = o_t \odot \tanh C_t \tag{15c}$$

where W is a weight matrix, b is a bias vector, subscripts denote step and superscripts denote hidden layer.

As we are dealing with a *many to one* classification problem, the output layer is computed by:

$$\hat{y} = W_{hy} \cdot h_T^L + b_y \tag{16}$$

where L is the number of hidden layers. The additional hyperparameters given in Table IV complete the LSTM network model structure.

TABLE IV
LSTM NETWORK MODEL HYPERPARAMETERS

Hyperparameter	Value
Number of inputs	11
Number of steps	128
Hidden layer size	32
Number of hidden layers	2
Number of outputs	7

The LSTM network model formed by the Eqs. 14, 15 and 16 and the model hyperparameters of the Table IV has 8 parameters with 17255 adjustable entries. Optimum values of these adjustable parameters were obtained by minimizing the following regularized risk function[6]:

$$J(\theta_k) \coloneqq \frac{1}{B} \sum_{j=1}^{B} D\{y_j, \operatorname{softmax}\left[\hat{y}(x_j, \theta_k)\right]\} + \frac{\lambda}{2} \sum_k \|\theta_k\|_F^2 \tag{17}$$

where $\theta_k \in \Theta$ are the LSTM network parameters, $x_j \in X$ are the training instances and $y_j \in Y$ are their corresponding ground truth labels[7]. Moreover, $D\{\cdot\}$ is the cross-entropy function, $\hat{y}(\cdot)$ is the network output, $\|\cdot\|_F$ is the Frobenius norm, B is the mini-batch size and $\lambda > 0$ is the so-called regularization parameter.

The regularized risk function given by Eq. 17 was minimized using *Adam* [43], an algorithm for efficient stochastic optimization that only requires first-order gradients with little memory requirement, with the mini-batch gradient descent hyperparameters listed in Table V.

[4]Which corresponds to almost 3 years of plant operation.

[5]Not all faults are listed because in this preliminary study we focus only in a small subset.

[6]Inspired by [42, p. 89].

[7]Both the ground truth label and the network output are one-hot-encoded.

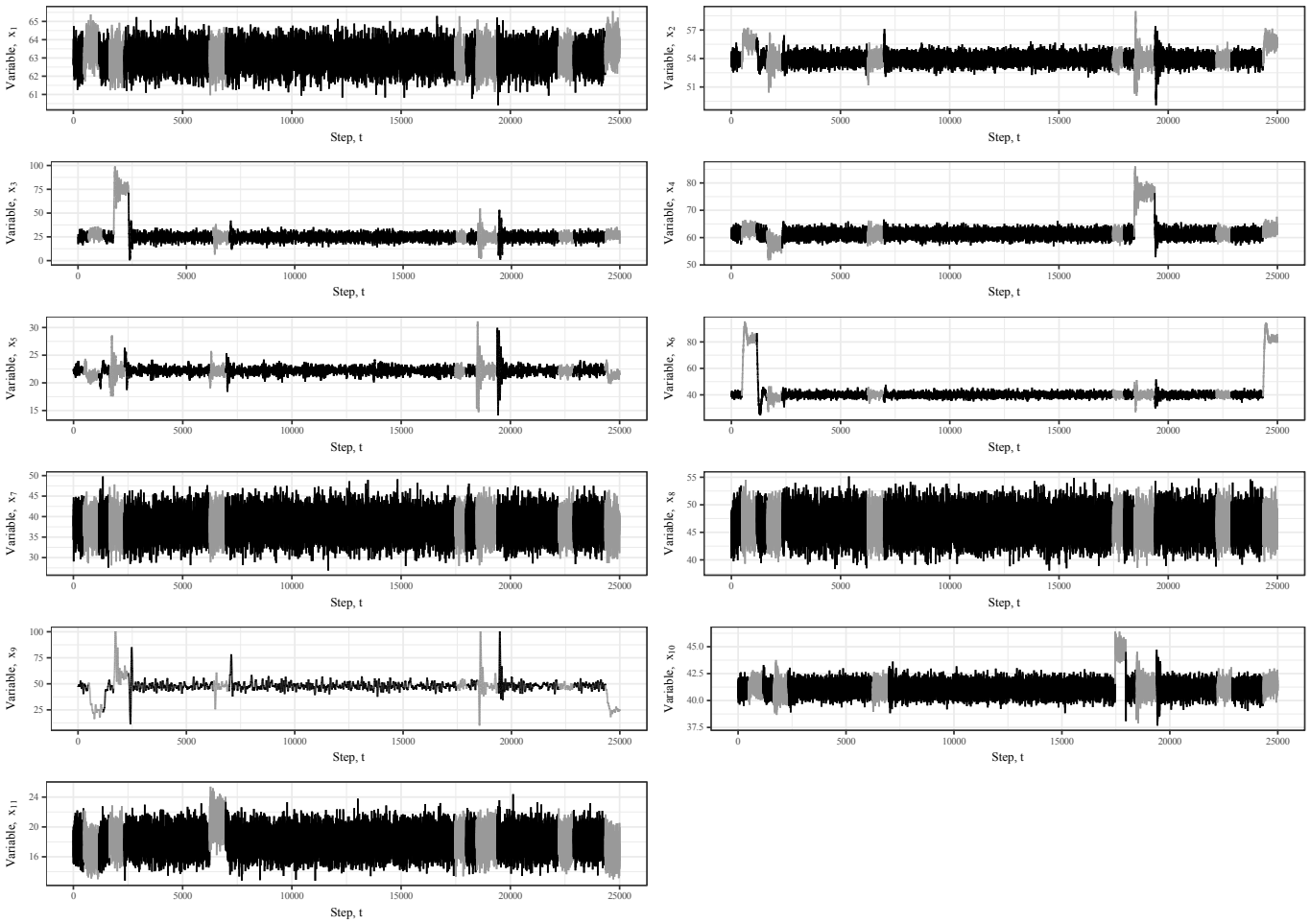

Fig. 3. TE-process simulated process history sample. The gray colored periods indicate fault presence.

TABLE V
MINI-BATCH GRADIENT DESCENT HYPERPARAMETERS

Hyperparameter	Value
Mini-batch size	1500
Regularization parameter	0.0015
Learning rate	0.0025
Number of training iterations	300×5733

C. Result comparisons

The performance of the proposed LSTM model was compared against fourteen data-driven FDD methods already published in the literature. From [30] we got results for two topological methods, namely: one map self-organizing maps (1-SOM) and multiple SOM (MSOM).

Results of nine FDD methods, such as principal component analysis (PCA), dynamic PCA (DPCA) independent component analysis (ICA), modified ICA (MICA), fisher discriminant analysis (FDA), partial least squares (PLS), total PLS (TPLS), modified PLS (MPLS) and subspace aided approach (SAP) were taken from [28].

Finally, from [27] we picked up results of three boosting methods: AdaBoost (ADA), de-noising boosting (DEN) and delay minimization and de-noising boosting (DEM). The performance comparison measure is the fault detection rate defined by the following equation:

$$DR_n = 100\frac{TP_n}{CP_n} \tag{18}$$

where TP_n is the number of fault n sequences correctly labeled by the FDD method, CP_n is the number of fault n sequences in the data set and n is the fault tag number ($n \in \{1, 2, 3, ..., 7\}$). Table VI shows the test set confusion matrix used to compute the fault detection rates of the proposed LSTM-based approach.

The fault detection rates of the novel LSTM FDD method were compared with the results from [30, Table 2, p. 13], [28, Table 7, p. 1576] and [27] using a 5-fold blocked cross-validation procedure [44, Figure 3.3, p. 37] and one-sample Wilcoxon signed rank tests[8] [45, p. 40]. Considering a confidence level of 95%, Table VII shows that our

[8]The alternative hypothesis was that our result is greater than the literature result.

TABLE VI
CONFUSION MATRIX

Ground truth label	Network output						
	$\hat{y}_0$	$\hat{y}_1$	$\hat{y}_2$	$\hat{y}_3$	$\hat{y}_4$	$\hat{y}_5$	$\hat{y}_7$
y_0	2007	0	0	22	1	0	0
y_1	0	65	0	0	0	0	0
y_2	0	0	63	0	0	0	0
y_3	52	0	0	23	0	0	0
y_4	0	0	0	0	54	0	0
y_5	0	0	0	0	0	68	0
y_7	0	0	0	0	0	0	103

approach outperforms 13 out of 14 already published linear and nonlinear FDD methods for fault y_3, which is a notable hard one according to the literature. Moreover, it is worth to note that to solve the FDD problem, the methods from the literature consume data from all 52 TE-process variables, while our novel deep-learning approach devours data from only 11 manipulated variables.

V. CONCLUSION

A novel LSTM-based FDD method was proposed and the simulated TE-process history of 11 manipulated variables, which includes 6 faults, was used to evaluate its performance against 14 data-driven FDD methods from the literature.

Unlike other TE-process data sets with the same faults, the proposed data set represents better the situation faced by engineers in the development of FDD systems, since it is quite unbalanced (i.e. it has fewer abnormal cases compared to the normal ones).

The results presented in Subsection IV-C show that the new paradigm proposed here has advantages over classical data-driven FDD methods as well new deep-learning approaches. First, it doesn't need any tunable detection threshold or even performs sequence-to-sequence prediction in order to detect faults as in [11]–[14]. Second, it uses data from only 11 variables from the whole set of 52 TE-process variables and third, it outperforms 13 out of 14 already FDD methods in the case of one of the most difficult, namely the fault y_3.

Several points like the evaluation of the complete set of faults, using the same data set to evaluate all methods, tuning the hyperparameters as well as the evalutation of other deep-learning methods (e.g. gated recurrent networks, convolutional neural networks) have not been addressed here and will be tackled in a future work.

Finally, for the sake of reproducibility, all the data and the code used in the elaboration of this paper are made publicly available [37].

ACKNOWLEDGMENT

The authors thank Mr. Christopher Olah who kindly gave us the permission to reproduce here a figure already published in his blog, the anonymous reviewers for their detailed and helpful comments to the manuscript and Petrobras for the permission to publish.

REFERENCES

[1] D. Himmelblau, *Fault Detection and Diagnosis in Chemical and Petrochemical Processes*. Amsterdam: Elsevier, 1978.
[2] V. A. Sutherland, M. Ehrlich, R. Engler, and K. Kulinowski, "Executive summary of explosion and fire at the Macondo well," US Chemical Safety and Hazard Investigation Board, Washington, DC, Investigation report 2010-10-I-OS, April 2016.
[3] S.-L. Jämsä-Jounela, "Future trends in process automation," *Annual Reviews in Control*, vol. 31, no. 2, pp. 211–220, 2007.
[4] E. L. Russell, L. H. Chiang, and R. D. Braatz, "Fault detection in industrial processes using canonical variate analysis and dynamic principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 51, no. 1, pp. 81–93, 2000.
[5] V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. N. Kavuri, "A review of process fault detection and diagnosis: Part I: Quantitative model-based methods," *Computers & Chemical Engineering*, vol. 27, no. 3, pp. 293–311, 2003.
[6] N. Klafke, M. B. d. Souza Jr., and A. R. Secchi, "A new benchmark for plantwide process control," *Brazilian Journal of Chemical Engineering*, vol. 33, no. 4, pp. 985–1002, 2016.
[7] L. Chiang, E. Russell, and R. Braatz, *Fault Detection and Diagnosis in Industrial Systems*. London: Springer, 2001.
[8] X. Dai and Z. Gao, "From model, signal to knowledge: A data-driven perspective of fault detection and diagnosis," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 4, pp. 2226–2238, 2013.
[9] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[11] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," in *23rd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, April 2015.
[12] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "LSTM-based encoder-decoder for multi-sensor anomaly detection," *arXiv preprint arXiv:1607.00148*, 2016.
[13] P. Filonov, A. Lavrentyev, and A. Vorontsov, "Multivariate industrial time series with cyber-attack simulation: Fault detection using an LSTM-based predictive data model," *arXiv preprint arXiv:1612.06676*, 2016.
[14] P. Filonov, F. Kitashov, and A. Lavrentyev, "RNN-based early cyber-attack detection for the Tennessee Eastman Process," *arXiv preprint arXiv:1709.02232*, 2017.
[15] D. Rumelhart, G. Hinton, and R. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.
[16] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.
[17] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.
[18] Krenker A., Beter J. and Kos A., *Artificial Neural Networks - Methodological Advances and Biomedical Applications*. Rijeka, Croatia: InTech, 2011, ch. Introduction to the Artificial Neural Networks, pp. 3–18.
[19] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.
[20] L. Fei-Fei, A. Karpathy, and J. Johnson, "Recurrent neural networks," http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf, 2017.
[21] A. Karpathy, J. Johnson, and L. Fei-Fei, "Visualizing and understanding recurrent networks," *arXiv preprint arXiv:1506.02078*, 2015.
[22] C. Olah, "Understanding LSTM networks," http://colah.github.io/posts/2015-08-Understanding-LSTMs, 2015.
[23] F. A. Gers, F. Cummins, and J. Schmidhuber, "Learning to forget: Continual prediction with LSTM," *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, 2000.
[24] A. Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*, ser. Studies in Computational Intelligence. Berlin: Springer, 2012.
[25] J. J. Downs and E. F. Vogel, "A plant-wide industrial process control problem," in *AIChE 1990 Annual Meeting, Session on Industrial Challenge Problems in Process Control, Paper 24a*, Chicago, Illinois, Nov. 14 1990.

TABLE VII
PERFORMANCE COMPARISON

Fault	Fault detection and diagnosis method														
	LSTM	**1-SOM**	**MSOM**	**PCA**	**DPCA**	**ICA**	**MICA**	**FDA**	**PLS**	**TPLS**	**MPLS**	**SAP**	**ADA**	**DEN**	**DEM**
y_1	100	**98**[a]	**99.9**[a]	100[g]	100[g]	**99.88**[a]	100[g]	100[g]	100[g]	100[g]	100[g]	**99.63**[a]	**99.19**[a]	**99.77**[a]	**99.8**[a]
y_2	99.62	**90.5**[a]	**97.8**[a]	99.38[d]	99.25[d]	98.75[d]	**98.38**[c]	**98.75**[c]	**98.88**[c]	**98.88**[c]	**98.88**[c]	**98.5**[c]	**97.1**[a]	**98.35**[c]	**98.41**[c]
y_3	37.04	34.5[d]	**10.8**[b]	**10.25**[b]	**23.25**[b]	**5**[b]	**13.25**[b]	**7**[b]	**11.75**[b]	**21.88**[b]	**18.75**[b]	**3.13**[b]	**1.51**[b]	**1.13**[b]	**1.29**[b]
y_4	97.82	97.4[d]	99.9[f]	100[g]	100[g]	100[g]	**93.13**[b]	100[g]	100[g]	100[g]	100[g]	99.63[f]	97.69[f]	99.91[f]	99.82[f]
y_5	99.65	**21.7**[a]	**5.9**[a]	**34.75**[a]	**67.75**[a]	100[g]	100[g]	100[g]	100[g]	100[g]	100[g]	100[g]	**87.07**[a]	99.27[d]	**98.98**[c]
y_7	99.66	99.7[d]	100[g]	100[g]	100[g]	100[g]	100[g]	100[g]	100[g]	100[g]	100[g]	100[g]	**97.92**[a]	99.75[d]	99.46[d]

[a] $p = 0.02$, [b] $p = 0.03$, [c] $p = 0.04$, [d] $p = 0.3$, [e] $p = 0.6$, [f] $p = 0.9$, [g] $p = 1$

[26] ——, "A plant-wide industrial process control problem," *Computers & Chemical Engineering*, vol. 17, no. 3, pp. 245–255, 1993.

[27] M. Grbovic, S. Vucetic, W. Li, P. Xu, and A. K. Usadi, "A boosting method for process fault detection with detection delay reduction and label denoising," in *Proceedings of the First International Workshop on Data Mining for Service and Maintenance*, ser. KDD4Service '11. New York: ACM, 2011, pp. 7–11.

[28] S. Yin, S. X. Ding, A. Haghani, H. Hao, and P. Zhang, "A comparison study of basic data-driven fault diagnosis and process monitoring methods on the benchmark Tennessee Eastman process," *Journal of Process Control*, vol. 22, no. 9, pp. 1567–1581, 2012.

[29] T. J. Rato and M. S. Reis, "Fault detection in the Tennessee Eastman benchmark process using dynamic principal components analysis based on decorrelated residuals (DPCA-DR)," *Chemometrics and Intelligent Laboratory Systems*, vol. 125, pp. 101–108, 2013.

[30] G. Robertson, M. C. Thomas, and J. A. Romagnoli, "Topological preservation techniques for nonlinear process monitoring," *Computers & Chemical Engineering*, vol. 76, pp. 1–16, 2015.

[31] J. C. M. Oliveira, K. V. Pontes, I. Sartori, and M. Embiruçu, "Fault detection and diagnosis in dynamic systems using weightless neural networks," *Expert Systems with Applications*, vol. 84, pp. 200–219, 2017.

[32] J. Zhu, Z. Ge, Z. Song, L. Zhou, and G. Chen, "Large-scale plant-wide process modeling and hierarchical monitoring: A distributed Bayesian network approach," *Journal of Process Control*, no. 10, 2017.

[33] P. R. Lyman, "Plant-wide control structures for the Tennessee Eastman Process," Master's thesis, Lehigh University, Bethlehem, Pennsylvania, 1992.

[34] P. R. Lyman and C. Georgakis, "Plant-wide control of the Tennessee Eastman Process," *Computers & Chemical Engineering*, vol. 19, no. 3, pp. 321–331, 1995.

[35] E. Russell, L. Chiang, and R. Braatz, *Data-driven Methods for Fault Detection and Diagnosis in Chemical Processes*. London: Springer, 2000.

[36] L. H. Chiang, E. L. Russell, and R. D. Braatz, "Fault diagnosis in chemical processes using fisher discriminant analysis, discriminant partial least squares, and principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 50, no. 2, pp. 243–252, 2000.

[37] G. M. Xavier, "TEP meets LSTM," https://github.com/gmxavier/TEP-meets-LSTM, 2018.

[38] A. Damien, "TensorFlow examples," https://github.com/aymericdamien/TensorFlow-Examples, 2016.

[39] G. Chevalier, "LSTMs for human activity recognition," https://github.com/guillaume-chevalier/LSTM-Human-Activity-Recognition, 2016.

[40] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[41] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *arXiv preprint arXiv:1409.2329*, 2014.

[42] B. Schölkopf and A. J. Smola, *Learning with Kernels – Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.

[43] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[44] C. N. Bergmeir, "New approaches in time series forecasting: methods, software and evaluation procedures," Ph.D. dissertation, Universidad de Granada, Granada, January 2013.

[45] M. Hollander and D. A. Wolfe, *Nonparametric Statistical Methods*. Hoboken: Wiley, 2014.