DETECTION OF OPERATIONAL FAILURES WITH ARTIFICIAL NEURAL NETWORKS: APPLICATION TO THE TENNESSEE EASTMAN PROCESS*

*Giovanni Morales-Medina***
*Sebastian Reyes-Angarita****

Received: 03/12/2023 • Accepted: 05/01/2024
https://doi.org/10.22395/rium.v23n44a1

ABSTRACT

The purpose of this article is to compare results of fault detection for the Tennessee Eastman (TE) process with the application of artificial neural networks (ANN). The Neuralnet library of the open-source program R, as well as the Keras library of the open-source program Python were used for the training of ANN. The TE process simulation data were downloaded from Harvard University's server, and subsequently analyzed, defining the trends in the operational variables during the appearance of failures. With the database, the training and validation of different ANN structures were developed, considering the parameters number of hidden neurons, activation function, and number of hidden layers. According to the results, the training and validation of the ANNs with the Neuralnet library yielded a lower performance in fault detection than that obtained with the Keras library. The ANN with the best performance in detecting failures in the TE process was obtained by the application of the Keras library. This ANN considered 52 input variables, 11 neurons in the hidden layer, and one neuron in the output layer, using a logistic function (ANN represented as 52:11:1 logistic) and reporting a prediction efficiency of 92% for the detection of faults with an external test set, which is convenient for future implementation in industrial processes.

Keywords: Artificial neural network, Tennessee Eastman process, faults, fault detection, fault prediction, process data, training of ANN, neuralnet, Keras.

* Funded by Universidad Industrial de Santander – project #3777 "Artificial neural networks in fault detection and diagnosis: performance evaluation in the Tennessee Eastman process".

** Escuela de Ingeniería Química, Facultad de Ingenierías Fisicoquímicas, Universidad Industrial de Santander, Bucaramanga – Colombia (E-mail: gmorales@uis.edu.co) Orcid: https://orcid.org/0009-0006-5215-0512

*** Escuela de Ingeniería Química, Facultad de Ingenierías Fisicoquímicas, Universidad Industrial de Santander, Bucaramanga – Colombia (E-mail: sebastian.reyes2@correo.uis.edu.co) Orcid: https://orcid.org/0009-0006-2384-3408

DETECCIÓN DE FALLAS OPERACIONALES CON REDES NEURONALES ARTIFICIALES: APLICACIÓN DEL PROCESO TENNESSEE EASTMAN

RESUMEN

Este artículo tiene como finalidad la comparación de resultados de detección de fallas en el proceso Tennessee Eastman (TE) con redes neuronales artificiales (RNA), utilizando las librerías neuralnet del programa de código abierto R y Keras del programa de código abierto Python. Para esto, los datos de la simulación de proceso TE fueron descargados del servidor de la universidad de Harvard, y posteriormente analizados, definiendo las tendencias en las variables operacionales ante las respectivas fallas. Con la base de datos, el entrenamiento y la validación de diferentes estructuras de RNA fue desarrollado considerando los parámetros: número de neuronas ocultas, función de activación y número de capas ocultas. Según los resultados, el entrenamiento y la validación de las RNA con la librería neuralnet reportó menores desempeños de detección de fallas, que las obtenidas con la librería Keras. La RNA de mejor desempeño en la detección de fallas del proceso TE correspondió a la estructura 52 variables de entrada, 11 neuronas en la capa oculta y una neurona en la capa de salida, con función logística y entrenada con la librería Keras (RNA representada como 52:11:1 *logistic*). Esta RNA presenta una eficiencia en la predicción del 92% para la detección de fallas en un conjunto externo de prueba, lo que resulta conveniente en una futura implementación en procesos industriales.

Keywords: Red neuronal artificial, proceso Tennessee Eastman, fallas, detección de fallas, predicción de fallas, datos de proceso, entrenamiento de RNA, neuralnet, Keras.

1. INTRODUCTION

Devices and equipment of industrial processes undergo failures, which can cause safety events with consequences on different levels, ranging from the expulsion of particulate matter and plant shutdowns to fatalities and environmental damage [1] [2]. Failures can occur due to a lack of control tuning and sensor calibrations, problems in actuators and motors, and fracture of materials, among others [3]. In the event of failures, process variables react following characteristic trajectories (temporal sequences of data) that can be used for developing detection models and identifying the root cause [4] [5]. Different detection models based on process data or simulations have been used to correct or mitigate safety events by directly intervening in the root cause of the failure [3] [6] [7]. These models have generally been developed considering projection methods and artificial intelligence (AI) algorithms [8] [9] [10] [11]. The former methods include inferential statistics and clustering such as principal components (PCA), partial least squares regression (PLS), and discriminant analysis (DA). These methods have shown limited performance for fault detection in highly non-linear processes, which usually exhibit high variable correlations and variable oscillations that do not distinguish between stationary and dynamic states [9] [10] [12]. For its part, the most representative AI method corresponds to artificial neural networks (ANN), which have shown adequate performance in highly non-linear processes and with correlation between variables [10]. Lei *et al*. [13] reviewed the different AI models applied to fault detection from a historical perspective. Regarding previous work in the area, several authors have implemented fault detection with a single ANN architecture [14] [15] [16]. Likewise, the influence of the number of internal layers and the number of neurons in these layers on the ANN performance in fault diagnosis has been analyzed, reporting an improvement in performance by increasing the number of hidden neurons and a setback in performance by increasing the number of failures in simultaneous predictions [14] [16] [17]. The selection of input variables for the ANN has commonly been established according to their level of correlation with the failure analyzed [15]. On the other hand, the values for the ANN output variables have been established as binary, with 0 for samples in normal operation and 1 for samples with failure [16] [17] [18] [19]. Additionally, a few studies have evaluated the performance of available program codes used in the training and validation of ANN [20] [21]. Mainly for reasons of industrial interest, computer programs with major applications in fault detection are commercial programs (C/C++, MATLAB, and LabView) [8] [22]. Alternatively, training and validation of ANNs can be developed through programs written in open-source code [8]; in fact, the academic community would see its interaction and research increase with further development of open-source fault detection algorithms [8]. These programs include R [23] [24] and Python [6] [21] [25] [26]. R is a language oriented to statistical analysis, and its ANN library called neuralnet has been used

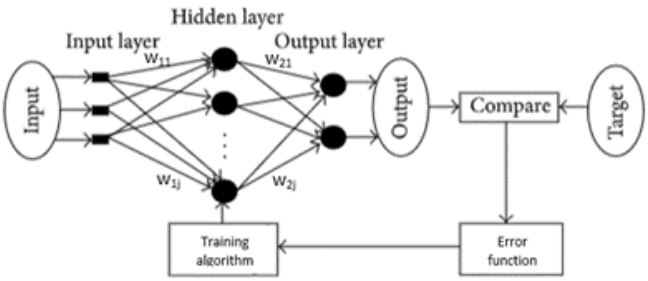
in failure prediction [24], process optimization [27], and process safety [28]. For its part, Python is a multipurpose high-level language, whose ANN library called Keras has been widely applied in fault detection [7] [21] [25]. The performances of ANN libraries from R, Python, and other programming environments on regression problems were compared by Fernández-Delgado *et al*. [29]. According to these authors, in general terms, Python libraries have better prediction performance than R. However, no comparisons have been found between the prediction performances of R and Python codes for fault detection. Considering the above and to motivate the application of open-source algorithms, this document presents a comparison of results between the neuralnet library of R and the Keras library of Python, applied to fault detection using the simulation data of the Tennessee Eastman process (TEP). Simulations of TEP have been used as benchmarking for failure models as an alternative to the limited access to industrial data [7] [8] [9]. In the following, Section 2 describes the methods related to the ANNs and the TE process, as well as the details of the analysis of simulation data and the application of the neuralnet and Keras libraries. Section 3 presents an examination of the trends of some of the failures of the TEP and the training and validation results from each library. The comparison between the performances of the libraries is detailed in Section 4.

2. MATERIALS AND METHODS

2.1 Artificial Neural Networks (ANN)

An ANN is a mathematical model that imitates the functioning of the human nervous system for information processing. Like the nervous system, an ANN also has interconnected neurons organized in levels (layers). The interconnections define the flow of information, determining the type of RNA [30]. Figure 1 illustrates the structure of an ANN named feedforward (unidirectional flow). According to this figure, neurons in a feedforward ANN are organized into three layers: an input layer that receives external information, a hidden layer that modifies the signals coming from the input layer, and an output layer that transfers data to the external devices of the ANN [14] [31]. The hidden layer can comprise more than one sublayer [14] [30] [31].

Figure 1. Representation of a feedforward ANN. Source: Yadav and Dash, 2014.



Each input signal to the neurons is multiplied by an associated weight (w). The sum of the product between inputs and weights enters the neurons, which apply an activation function, generating the respective output signals. The learning or training of an ANN consists of the application of an algorithm that defines the values of the weights of the connection signals (Figure 1) [14]. The algorithm called backpropagation is the most used training algorithm, since it presents information about the sensitivity of the network's predictions concerning the weights of the signals. The algorithm updates the weights based on the gradient of an error function according to the Levenberg-Marquardt optimization method [14] [31].

2.2 Tennessee Eastman Process (TEP)

The simulation of the hypothetical process called Tennessee Eastman (TEP), created in 1993 by the company Eastman Chemical [32], is usually referenced as benchmarking for the analysis of the performance of fault detection and control algorithms [7] [18] [33] [34] [35] [36]. The TEP is used as a benchmark because it contains a module that enables the scheduling of the appearance of up to 22 types of faults and disturbances, programmed in a staggered manner or simultaneously, as well as fault-free operation [18] [37]. In brief, the TEP is composed of a reactor, a condenser, a liquid/vapor separator, a compressor, and a distillation column or stripper. This process is illustrated in the diagram in Figure 2. The TEP considers eight components, two main reactions (1 and 2), and two secondary reactions (3 and 4). The four reactions have exothermic and irreversible characteristics [18].

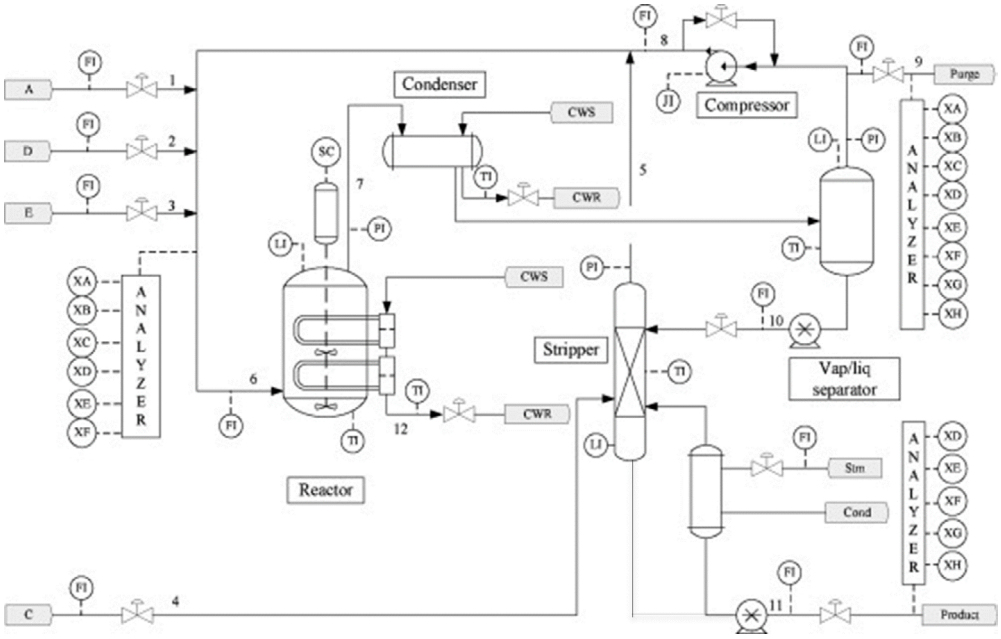
$$A_{gas} + C_{gas} + D_{gas} \rightarrow G_{liquid} \tag{1}$$

$$A_{gas} + C_{gas} + E_{gas} \rightarrow H_{liquid} \tag{2}$$

$$A_{gas} + E_{gas} \rightarrow F_{liquid} \tag{3}$$

$$3D_{gas} \rightarrow 2F_{liquid} \tag{4}$$

Figure 2. Diagram of the Tennessee Eastman process. Source: Yin *et al*., 2012.



In the operation, feed streams 1, 2, and 3 are the flows of the reactants in the gas phases named A, D, and E, respectively. These streams are mixed with two recirculation streams (5 and 8), forming stream 6, which is directed to the reactor (Figure 2). The reactor output flow is partially condensed and sent to the separator. The liquid flow from the separator (stream 10) is fed to the upper plate of the stripper, while the vapor from the separator (stream 8) is recirculated to the reactor through a compressor. In recirculation, a fraction of gas is purged (stream 9), avoiding the accumulation of the byproduct F and the inert B. Similarly, in the stripper, there are two main sources of steam: firstly, a reboiler and, secondly, the feed of reagent C (stream 4). The main products of the process are generated from the stripper (stream 11) [34] [36].

2.3 TEP Simulation Data

The simulation data of the TEP were downloaded from the database of Harvard University [37] and analyzed, defining trends in the variables with the appearance of failures. This database was made up of 12 manipulated variables and 41 dependent or output variables, with samples taken every 3 minutes for 48 hours of simulation time. Likewise, the database comprised 16 known failures and 5 unknown ones, for a total of 21 failures. During simulation, those failures appeared in the inlet flows, the

reactor, the condenser, and the valves (sticking). A new variable called "operation" was included in the database, considering a value of 1 for samples in failure and a value of 0 for samples in normal operation. Likewise, the database variables were normalized between 0 and 1 for use in the ANN training and validation codes.

2.4 Training and Validation Using neuralnet

The neuralnet library of R was used in the training and validation of feedforward ANN architectures. Neuralnet is a package that allows the flexible configuration of multiple hyperparameters and the generalized classification of these to optimize their performance. This library has algorithms such as backpropagation (backprop) and elastic backpropagation (rprop+), which update the network weights. In the execution of the library, the sigmoid activation functions (logistic) and hyperbolic tangent (tanh) were established for the hidden layer neurons (the code fragment can be viewed in Appendix A). Likewise, for the output neurons, the linear activation function was used. The minimization function used in training corresponded to the mean square error, according to,

$$MSE = \frac{1}{M}\sum_{i=1}^{M}\left(Target_i - Predicted_i\right)^2 \tag{5}$$

Where, $Target_i$ and $Predicted_i$ correspond to the value of the variable "operation" and the value predicted by the ANN, respectively. M is the number of predictions or samples. The training and validations using neuralnet were carried out in RStudio.

2.5 Training and Validation in Keras

The Keras library was used with the Adam training optimization code. Keras applies the training through deep learning models. The main parameters in Keras correspond to the loss function, the optimizers, the activation function, and the network architecture. As for neuralnet, the minimization or loss function in training corresponds to the MSE (equation 5). Similarly, overfitting of network training can be checked using additional data for validation of network learning performance. The Google Colab editor was used in the coding of a routine to carry out the training and validations with the Python libraries. The code fragment can be checked in Appendix B.

The ANN architectures were proposed considering the following parameters: number of neurons, activation function, number of hidden layers, and tolerance for convergence. The training of the ANN architectures was executed with 70% of the data, while the validation was carried out with the remaining data. The ANN with the lowest MSE parameter (calculated with the average between validation and training) was selected as the best network for fault detection in the TE process.

3. RESULTS

3.1 Tennessee Eastman Process Data

The simulation database reported a total of 9,850,000 samples and 52 variables [37]. This base refers to 250,000 samples in normal operation and the rest in fault operation (*i.e*. 480,000 samples for each failure). The known faults codified in the simulation and reported in the database are described in Table 1. The simulation presents 15 types of failures: three related to compositions (failures 1, 2, and 8), seven related to temperatures (failures 3, 4, 5, 9–12), four related to flows (failures 6, 7, 14, and 15), and one related to reaction kinetics (failure 13). Additionally, the simulation included 5 failures of unknown origin [37].

Table 1. Known failures in the Tennessee Eastman process [32]

Failure	Disturbance	Class
1	A/C ratio. Composition of constant B (current 4)	Step
2	B composition, constant A/C ratio (current 4)	Step
3	Feed temperature D (stream 2)	Step
4	Cooling water temperature at the reactor inlet	Step
5	Condenser cooling water temperature	Step
6	Loss of power A (current 1)	Step
7	Pressure loss at C, flow reduction in stream 4	Step
8	Power composition A, B, C (stream 4)	Random
9	Feed temperature D (stream 2)	Random
10	Feed temperature C (stream 4)	Random
11	Cooling water temperature at the reactor inlet	Random
12	Temperature of cooling water at the condenser inlet	Random
13	Reaction kinetics	Decrease
14	Water valve for reactor cooling	Opening
15	Condenser cooling water valve	Opening

Table 2 shows the limits for normal operation, according to the database, for the reactor, the separator, and the distillation column. In addition to the normal process limits, this table reports the limits established in the simulation for operation shutdown. The upper shutdown limit taken for the reactor, the separator, and the column levels corresponded to the full volume of the equipment. Regarding concentrations, normal operation is considered to be in a fluctuation range of ±5% w [32].

Table 2. Limits for normal operation and shutdown of the TEP

Variable	Operating limits		System shutdown limits	
	Lower	Upper	Lower	Upper
Reactor pressure	None	2895 kPa	None	3000 kPa
Reactor level	50% (10.65 m^3)	100% (21.3 m^3)	2.0 m^3	24 m^3
Reactor Temperature	None	150°C	None	175°C
Separator level	30% (3.0 m^3)	100% (9 m^3)	1.0 m^3	12.0 m^3
Column level	30% (1.98m^3)	100% (6.6 m^3)	1.0 m^3	8.0 m^3

On the other hand, Figure 3 shows the variations in the valve openings associated with flow A (XMV3) (stream 1) for failure 6 (Table 1), purge flow (XMV6) (Figure 2, stream 9) for failure 2 (Table 1), and cooling water flow to the reactor jacket (XMV10) (stream 12) for failure 4. The valve opening percentages are part of the simulation information. In Figure 3, the samples are shown in both normal operating conditions and faulty operating conditions. According to this figure, the valve openings present slight variations up to sample 200 that indicate pseudo-steady state operation (normal operation), while subsequent samples show fluctuations in the openings beyond the pseudo-steady state limits. Those fluctuations indicate the appearance of the respective failures. It is important to mention that changes in position of valve 12, when facing the reactor temperature elevation (failure 4), are small due to the high value of the respective valve coefficient. Due to this, a slight valve movement generates a high change in the cooling flow, leading to a prompt regulation of reactor temperature. The above is illustrated in Figure 4, which presents the fluctuations in reactor temperature upon the appearance of failure 4 and the activation of the respective control loop (cooling flow to the jacket). The action of the control loop is visualized in Figure 4 by the oscillations around the control point (120.4 °C) after sample 200. Accordingly, the variation in the control point is about ±0.3°C.

Figure 3. Variations in valve openings in the event of failures (see Figure 2). XMV3: flow A valve, stream 1. XMV6: purge valve, stream 9. XMV10: reactor jacket valve, stream 12. Source: own elaboration.

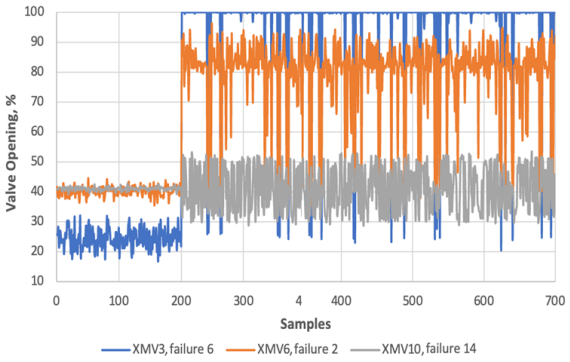
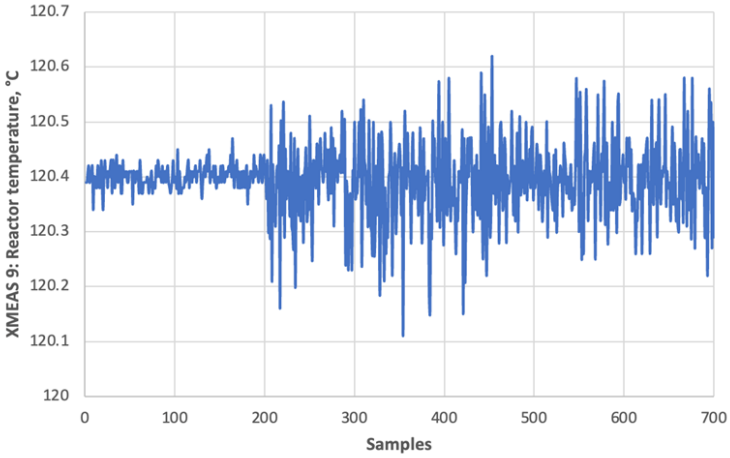


Figure 4. Reactor temperature variations (°C) upon failure 4 and activation of the control loop. Source: authors.



Each process failure is related to a set of variables [32]. The respective variables associated with each failure were identified by the comparison of the fluctuations of the samples in the database and the normal operating limits (Table 2). The variables associated with the failures were used for the detection through different ANN architectures. The results are presented below.

3.2 Training and Validation with the neuralnet Library.

The ANN architectures that considered logistic activation functions were described as "52:x:1 logistic". That is, 52 input neurons, x neurons in the inner layer, and one system prediction neuron in normal operation (value 0) or failure (value 1), for any type of failure (Table 1 plus 5 unknown failures), applying the logistic function for the hidden or internal neurons. The architectures were trained and validated with neuralnet considering a set of 500 samples, a threshold=0.2, and different numbers of internal neurons. These parameters were established due to the computational demand for executing RStudio. Figure 5 presents the variations in the training, validation, and total MSE, with the number of internal neurons. Likewise, Figure 6 shows the respective variations in the coefficient of determination R^2. From Figures 5 and 6, it is possible to infer that the training prediction improves with the number of internal neurons (*i.e.* decreasing the MSE and increasing the R^2). However, the results of the validation (data not used in training) decreased with the number of neurons in the hidden layer, indicating a specialization of the ANN in the reproduction of the training set (overfitting). According to the total or average MSE, the best prediction performance was achieved with 8 internal neurons (RNA 52:8:1 logistic), reporting a total MSE=0.052 and total R^2=0.67. From this, the application of neuralnet function resulted

in poor overall performance in fault detection in the TE process. Given this, different options for neuralnet were tested, considering the number of input data, threshold, and activation functions. Neither option led to a significant improvement in prediction performance with neuralnet.

Figure 5. Neuralnet results for MSE with 500 data, Logistic function, threshold=0.2 Source: authors.

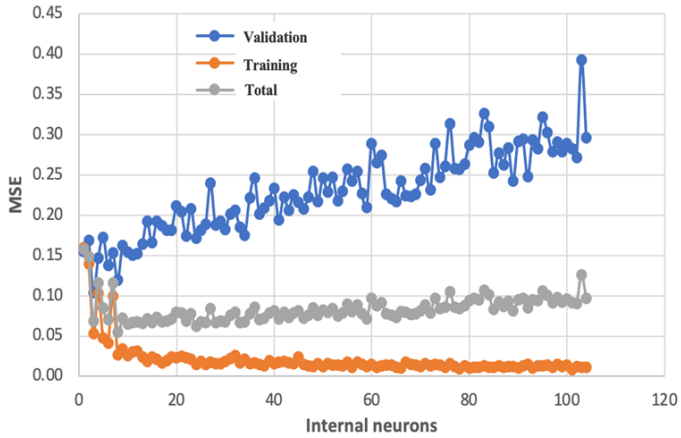
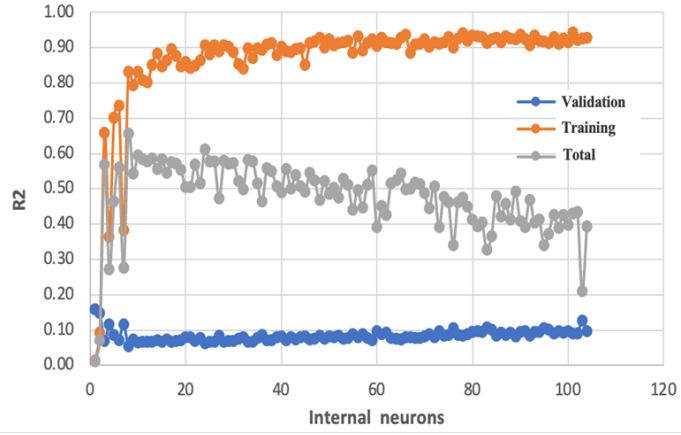


Figure 6. Neuralnet results for coefficient of determination with 500 data, Logistic function, threshold=0.2. Source: authors.



3.3 Training and Validation with the Keras Library

As with neuralnet, the Keras library execution considered the total number of failures of the TEP (Table 1 plus 5 unknown failures). Figure 7 defines the variation in the Total MSE with internal neurons (logistic function) in the detection of failing samples. Accordingly, it is possible to mention that the lowest total MSE is obtained with 11 hidden neurons (*i.e.* 52:11:1 logistic). Figure 8 presents the training MSE values and the

validation for this Keras ANN. It follows from this figure that the number of iterations leads to an improvement in both training and validation, ruling out overfitting or specialization of the ANN in the fault detection process. It is necessary to clarify that the Google Colab environment enabled Keras training with a set of 83,000 samples, of which 80,000 were samples in a state of failure (4,000 per failure). Furthermore, additional validations of the ANN were applied with 2 additional sets, with 4,800 samples per set (800 samples in normal operation and 200 for each failure). These validations confirmed the trends in Figure 7 for the 52:11:1 logistic ANN.

Figure 7. Keras results for the total MSE with 87,800 data, 52:x:1 logistic. Source: authors

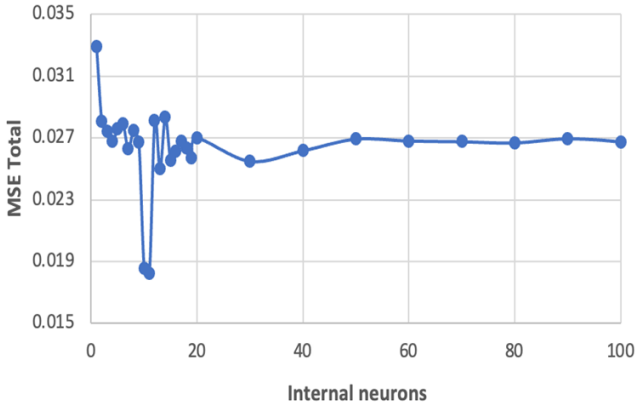
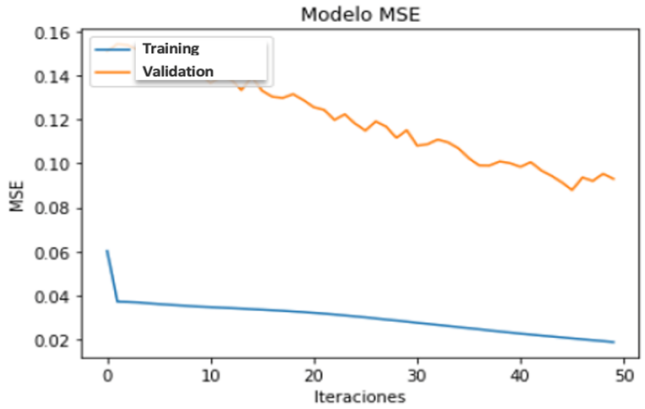


Figure 8. Keras results for training and validation iterations, 52:11:1 logistic. Source: authors



Besides, the number of hidden layers of the ANN was increased to two sublayers, defining the ANN structure "52:11:X:1 logistic". The number of neurons in the second layer (X) was varied, determining the prediction performance. Table 3 shows the total MSE values for the increment of the second-sublayer neurons. According to this table,

the values obtained for the total MSE with a second sublayer were higher than those reported with the ANN 52:11:1 logistic. Taking this into account, the performance of the ANN declined with the addition of a second sublayer.

Table 3. Verification of the number of hidden layers. Structure 52:11:X:1

Sublayers	No. Neurons		Total MSE
	Sublayer 1	Sublayer 2	
2	11	10	0.030
2	11	20	0.029
2	11	30	0.029
2	11	40	0.032
2	11	50	0.025

Additionally, the architecture of the ANN was augmented to consider 20 failures for the TEP (*i.e.* 52:x:21 logistic). With this, in addition to detection, the ANN can be used for the diagnosis of the type of failure. The output layer establishes a prediction for the type of failure plus the normal operating condition. Figure 9 presents the variation of the total MSE for the 52:x:21 logistic ANN with the number of hidden neurons. According to this figure, the hidden layer with 80 neurons led to the best prediction performance (*i.e.* 52:80:21 logistic). Similarly, the performance of the ANN 52:80:21 logistic was analyzed with the increase in the number of hidden sublayers and neurons. Table 4 presents the results of the performances with the addition of up to two hidden sublayers (x, z). According to this table, the best performance (total MSE=9.3) is achieved with three hidden sublayers, according to ANN 52:80:50:30:21 logistic. This total MSE indicates that the application of the Keras library leads to low performance when detection and diagnosis are carried out simultaneously for the TEP failures. This coincides with different literature reports [16] [18] [36] [37].

Figure 9. Keras results for the total MSE with 87,800 data, function 52:x:21 logistic. Source: authors.

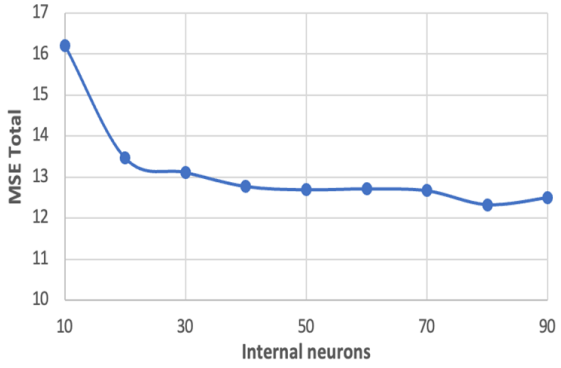


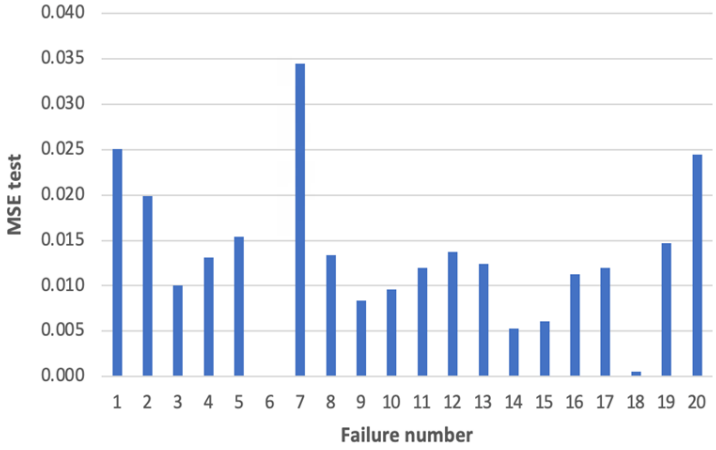
Table 4. Prediction performances with the number of hidden layers 52:80:x:z:21 logistic.

Sublayers	No. Neurons			Total MSE
	Sublayer 1	Sublayer 2	Sublayer 3	
1	80	-	-	12.3
2	80	20	-	11.6
2	80	30	-	12.3
2	80	40	-	11.7
2	80	50	-	11.1
3	80	50	20	10.1
3	80	50	30	9.3
3	80	50	40	9.5

4. ANALYSIS OF RESULTS

According to the results, the ANN 52:11:1 logistics trained with the Keras library of Python reported the lowest value of total MSE (MSE= 0.018), which is *ca* 3 times lower compared to the ANN 52:8:1 logistic trained with the neuralnet library of R (total MSE=0.05). The RNA 52:11:1 logistic was additionally tested with a set composed of 400 samples for each failure. Figure 10 shows the results of this ANN for each failure. From this figure, it is possible to mention that the test reported values of less than 0.035 for MSE, indicating satisfactory results from the trained ANN using the Keras library. Likewise, the best predictions occurring with failures 6 and 18 (test MSE of $5.64*10^{-09}$ and $5.06*10^{-04}$, respectively), referring to a decrease in the flow of feed A and an unknown failure (Table 1), respectively. An analysis of the trends of these faults showed high fluctuations, which suggested that the ANN detects those respective faults with a better performance. On the other hand, the predictions showing the lowest performance corresponded to those for failures 1, 2, 7, and 20 (Figure 10). These failures corresponded to fluctuations in the ratio of feed A to feed C, alterations in the composition of feed B, a decrease in flow in feed C, and an unknown failure, respectively. The fluctuation trends in these faults showed moderate amplitude, and, therefore, the performance results of ANN 52:11:1 logistics were lower than the performance for failures 6 and 18. Accordingly, the detection of fault appearance with the ANN 52:11:1 logistic showed an efficiency of prediction of 92% in the test with 8,000 samples.

Figure 10. Keras results for the test with the ANN 52:11:1 logistic. Source: author.



5. CONCLUSIONS

The ANNs trained with the neuralnet library showed a maximum detection performance with a total MSE of 0.05, resulting from training with a low sample set, limited by computational resources. For its part, training the ANNs with the Keras library reported higher detection performance, reducing the total MSE to 0.018, obtained with a set of 83,000 samples. The best prediction was reported by an ANN with 52 input neurons, 11 neurons in the hidden layer, and one neuron in the output layer (52:11:1 logistic). This ANN showed an efficiency of 92% when evaluated with a set of 8,000 samples for the individual detection of each failure. Likewise, an increase in the number of hidden layers was not favorable for the increase in the detection performance of this ANN. Additionally, the detection and diagnosis of the type of failure were analyzed with an ANN trained using the Keras library. The ANN with 52 input variables, three hidden layers (80, 50, and 30 hidden neurons, respectively), and 21 output variables (20 failures and normal operation) (*i.e.* 52:80:50:30:21 logistic) reported the lowest value for the total MSE (9.34). This performance value was insufficient for detection and diagnosis with an external set of samples.

RECOGNITIONS

To the Industrial University of Santander, for its financial support with project # 3777 "Redes neuronales artificiales en la detección y el diagnóstico de fallas: evaluación del desempeño en el proceso Tennessee Eastman".

REFERENCES

[1] J. A. Ramírez, H. O. Sarmiento, and J. M. López, "Diagnóstico de Fallas En Procesos Industriales Mediante Inteligencia Artificial". Revista Espacios, vol. 39, no. 24, pp.16, 2018.

[2] M. A. Turk, and A. Mishra. "Process safety management: Going beyond functional safety". Hydrocarbon Processing, vol. 92, no. 3, pp. 55-63, March 2013.

[3] R. Rivas, P. Zubieta, and H. Garcini, "Detección y Diagnóstico Automático de Fallos En Procesos Industriales". Ingeniería electrónica, Automática y Comunicaciones, volumen XXIII, no. 3, 2002.

[4] E. E. Tarifa, and S. L. Martinez. 2007, "Diagnostico de Fallas Con Redes Neuronales. Parte 1: Reconocimiento de Trayectorias". Revistas electrónicas UN ingeniería e Investigación, vol. 27, no.1, pp. 68-76, Abril 2007.

[5] H. Hassanpour, B. Corbett, and P. Mhaskar, "Artificial Neural Network-Based Model Predictive Control Using Correlated Data". Industrial & Engineering Chemistry Research, vol. 61, no. 8, pp. 3075–3090, 2022, https://doi.org/10.1021/acs.iecr.1c04339.

[6] Y. Shin, R Smith, and S. Hwang, "Development of model predictive control system using an artificial neural network: A case study with a distillation column". Journal of Cleaner Production, vol. 277, pp. 124124, 2020, https://doi.org/10.1016/j.jclepro.2020.124124.

[7] R. Arunthavanathan, F. Khan, S. Ahmed, S. Imtiaz, and R. Rusli, "Fault detection and diagnosis in process system using artificial intelligence-based cognitive technique". Computers & Chemical Engineering, vol. 134, pp. 106697, 2019, https://doi.org/10.1016/j.compchemeng.2019.106697.

[8] A. Melo, M.M. Câmara, J.C. Pinto, "Data-Driven Process Monitoring and Fault Diagnosis: A Comprehensive Survey". Processes, Vol. 12, no. 2, pp. 251, 2024, https://doi.org/10.3390/pr12020251

[9] S. Yin, S.X. Ding, A. Haghani, H. Hao, 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, https://doi.org/10.1016/j.jprocont.2012.06.009

[10] J. Yu, Y. Zhang, "Challenges and opportunities of deep learning-based process fault detection and diagnosis: a review". Neural Comput & Applic, Vol. 35, pp. 211–252, 2023, https://doi.org/10.1007/s00521-022-08017-3

[11] D. Divya, B. Marath, M.B. Santosh Kumar, "Review of fault detection techniques for predictive maintenance". Journal of Quality in Maintenance Engineering, Vol. 29, no. 2, pp. 420-441, 2023, https://doi.org/10.1108/JQME-10-2020-0107

[12] L.H. Chiang, E.L. Russell, 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, https://doi.org/10.1016/s0169-7439(99)00061-1

[13] Y. Lei, B. Yang, X. Jiang, F. Jia, N. Li, A.K. Nandi, "Applications of machine learning to machine fault diagnosis: A review and roadmap". Mechanical Systems and Signal Processing, Vol. 138, 106587, 2020, https://doi.org/10.1016/j.ymssp.2019.106587

[14] A. Yadav, and Y, Dash, "An overview of transmission line protection by artificial neural network: fault detection, fault classification, fault location, and fault direction discrimination". Advances in Artificial Neural Systems, vol. 1, pp. 6, 2014, http://dx.doi.org/10.1155/2014/230382

[15] J. Hong, J. Qu, W. Tian, Z. Cui, Z. Liu, Y. Lin, and C. Li, "Identification of Unknown Abnormal Conditions in Catalytic Cracking Process Based on Two-Step Clustering Analysis and Signed Directed Graph". Processes, vol. 9, no.11, pp. 2055, 2021, https://doi.org/10.3390/pr9112055

[16] G. Taira, S. Park, A. Zanin, and C. Porfirio, "Fault detection in a fluid catalytic cracking process using Bayesian recurrent neural network". IFAC PapersOnLine, vol. 55, no.7, pp. 715–720, 2022, https://doi.org/10.1016/j.ifacol.2022.07.528

[17] V. Venkatasubramanian, K. Chan, "A neural network methodology for process fault diagnosis". AIChE Journal, vol. 35, no 12, p. 1993-2002, 1989, https://doi.org/10.1002/aic.690351210

[18] L. Chiang, E. Russell, and R. Braatz, "Fault Detection and Diagnosis in Industrial Systems", Londres, Springer Science & Business Media, 2000, https://doi.org/10.1007/978-1-4471-0347-9

[19] J. Hong, J. Qu, W. Tian, Z. Cui, Z. Liu, Y. Lin, and C. Li, "Identification of Unknown Abnormal Conditions in Catalytic Cracking Process Based on Two-Step Clustering Analysis and Signed Directed Graph". Processes, vol. 9, no.11, pp. 2055, 2021, https://doi.org/10.3390/pr9112055

[20] U. Sarwar, M. Muhammad, A. A. Mokhtar, R. Khan, P. Behrani, S. Kaka, "Hybrid intelligence for enhanced fault detection and diagnosis for industrial gas turbine engine". Results in Engineering, vol. 21, 101841, 2024, https://doi.org/10.1016/j.rineng.2024.101841

[21] A.O. e Souza, M.B. de Souza, F.V. da Silva, "Enhancing fault detection and diagnosis systems for a chemical process: a study on convolutional neural networks and transfer learning". Evolving Systems, vol. 15, pp. 611-633, 2024, https://doi.org/10.1007/s12530-023-09523-y

[22] S. Lu, J. Lu, K. An, X. Wang, Q. He, "Edge Computing on IoT for Machine Signal Processing and Fault Diagnosis: A Review". IEEE Internet of Things Journal, vol. 10, no. 13, pp. 11093-11116, 2023, https://doi.org/10.1109/JIOT.2023.3239944

[23] F. Günther, and S. Fritsch. "Neuralnet: training of neural networks". R Journal, vol. 2, no. 1, pp. 30–38, 2010, https://doi.org/10.32614/RJ-2010-006

[24] G. Iannace, G. Ciaburro, and A. Trematerra, "Neural Networks Model to Detect Wind Turbine Dynamics". International Journal of Automation and Smart Technology, 2020, vol. 10, no 1, pp. 145-152, 2020, https://doi.org/10.5875/ausmt.v10i1.2225

[25] J.-L. Kang, "Visualization analysis for fault diagnosis in chemical processes using recurrent neural networks". Journal of the Taiwan Institute of Chemical Engineers, vol. 112, pp. 137-151, 2020, https://doi.org/10.1016/j.jtice.2020.06.016.

[26] S. Chen, J. Yu, and S. Wang, "One-dimensional convolutional auto-encoder-based feature learning for fault diagnosis of multivariate processes". Journal of Process Control, vol.87, pp.54–67, 2020, https://doi.org/10.1016/j.jprocont.2020.01.004.

[27] S. Sano, T. Kadowaki, K. Tsuda, and S. Kimura, "Application of Bayesian Optimization for Pharmaceutical Product Development". Journal of Pharmaceutical Innovation, vol. 15, pp. 333-343. 2020, https://doi.org/10.1007/s12247-019-09382-8.

[28] A. Zolfaghari, and M. Izadi, "Burst Pressure Prediction of Cylindrical Vessels Using Artificial Neural Network". Journal Pressure Vessel Technology, vol. 142, no. 3, pp. 031303, 2020, https://doi.org/10.1115/1.4045729.

[29] M. Fernández-Delgado, M. Sirsat, E. Cernadas, S. Alawadi, S. Barro, and M. Febrero-Bande, "An extensive experimental survey of regression methods". Neural Networks, vol. 111, pp. 11–34, 2019, https://doi.org/10.1016/j.neunet.2018.12.010

[30] H. González, and H. Martínez, "Redes Neuronales Artificiales: Fundamentos, modelos y aplicaciones", Madrid, 2000.

[31] P. Isasi, and I. Galván, "Redes de neuronas artificiales. Un Enfoque Práctico", Editorial Pearson Educación SA Madrid España, 2004.

[32] J. J. Downs, and E. Vogel, "A plant-wide industrial process control problem". Computers & Chemical Engineering, vol. 17, no.3, pp. 245–255, March 1993, https://doi.org/10.1016/0098-1354(93)80018-I.

[33] E. Andersen, I. Udugama, K. Gernaey, C. Bayer, and M. Kulahci, "Big Data Generation for Time Dependent Processes: The Tennessee Eastman Process for Generating Large Quantities of Process Data". Computer Aided Chemical Engineering , vol. 48, pp. 1309–1314, 2020, https://doi.org/10.1016/B978-0-12-823377-1.50219-6.

[34] S. Yin, S. 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, https://doi.org/10.1016/j.jprocont.2012.06.009.

[35] S. Heo, and J. Lee, "Fault detection and classification using artificial neural networks". IFAC-PapersOnLine, vol. 51, no. 18, pp. 470–475, 2018, https://doi.org/10.1016/j.ifacol.2018.09.380.

[36] N. Basha, C. Kravaris, H. Nounou, and M. Nounou, "Bayesian-optimized Gaussian process-based fault classification in industrial processes". Computers Chemical Engineering, vol. 170, pp. 108126, 2023, https://doi.org/10.1016/j.compchemeng.2022.108126.

[37] C. Rieth, B. Amsel, R Tran, and M. Cook, "Additional Tennessee Eastman ProcessSimulation Data for Anomaly Detection Evaluation". Harvard Dataverse: vol. 1, 2017, https://doi.org/10.7910/DVN/6C3JR1

APPENDIXES

Appendix A:

Figure 10 Coding of the RNA structure in R neuralnet package. Source: author.

```
> set.seed(nrow(data))
> i<-0
> MSEnn<-matrix(nrow=104,ncol=5)
> MSEnnt<-matrix(nrow=104,ncol=5)
> MSEnntot<-matrix(nrow=104,ncol=5)
> R2_<-matrix(nrow=104,ncol=5)
> R2_t<-matrix(nrow=104,ncol=5)
> R2_tot<-matrix(nrow=104,ncol=5)
> for (k in 1:104) {
      i=i+1
      for (j in 1:5) {
          index = sample(seq_len(nrow(scaled01)), size = samplesize)
          Dtrain<- scaled01[index,]
          Dtest <- scaled01[-index,]
          colnames(Dtrain) <- n
          colnames(Dtest) <- n
          f <- as.formula(paste("Output ~", paste(n[!n %in% "Output"], collapse = " + "
))))
          model0 <- neuralnet(f,data=Dtrain,hidden=c(k),linear.output=T,err.fct="sse"
,act.fct="logistic",stepmax=100000,threshold=0.2, algorithm="rprop+",rep=3)
          pr.nn <- compute(model0,Dtest[,1:52])
          pr.nn_ <- pr.nn$net.result*(max(data$Output)-min(data$Output))+min
(data$Output)
          test.r <- Dtest[,53]*(max(data$ Output)-min(data$ Output))+min(data$ Output)
          ym_<-mean(Dtest[,53])
          MSEnn[i,j]<-sum((test.r-pr.nn_)^2)/nrow(Dtest)
          R2_[i,j]<-1-sum((test.r-pr.nn_)^2)/sum((test.r-ym_)^2)
          pr.nn <- compute(model0,Dtrain[,1:52])
          pr.nn_t <- pr.nn$net.result*(max(data$ Output)-min(data$ Output))+min(data$
Output)
          test.rt<- Dtrain[,53]*(max(data$ Output)-min(data$ Output))+min(data$ Output)
          ym_t<-mean(Dtrain[,53])
          MSEnnt[i,j]<-sum((test.rt-pr.nn_t)^2)/nrow(Dtrain)
          R2_t[i,j]<-1-sum((test.rt-pr.nn_t)^2)/sum((test.rt-ym_t)^2)
          MSEnntot[i,j]=(sum((test.r-pr.nn_)^2)+sum((test.rt-pr.nn_t)^2))/(nrow(data))
          R2_tot[i,j]<-1-(sum((test.r-pr.nn_)^2)+sum((test.rt-pr.nn_t)^2))/(sum((test.r
-ym_)^2)+sum((test.rt-ym_t)^2))
      }
      print(k)
}
```

Appendix B:

Figure 11 Code fragment of the RNA 52:11:1
structure with the Keras package in Python. Source: author.

```
[ ] #Modelo de la RNA, definición de numero de capas, numero de neuronas por capa, número de interacción entre
    #los parámetros, función de activación, función de perdida y optimizador
    model=Sequential()
    model.add(Dense(52,input_dim=52,kernel_initializer='normal',activation='sigmoid'))
    model.add(Dense(11,input_dim=11,kernel_initializer='normal',activation='sigmoid'))
    model.add(Dense(1,input_dim=1,kernel_initializer='normal',activation='sigmoid'))
    model.compile(loss='mean_squared_error',optimizer="adam")

[ ] #Entrenamiento de la RNA definiendo el tamaño de muestra, número de iteraciones y datos de validación
    history=model.fit(x,y,batch_size=100,epochs=50,validation_data=(x_validation,y_validation))

[ ] #Evaluación del modelo de entrenamiento con datos de prueba
    resultado=model.evaluate(x_test,y_test)
    print("Evaluación MSE de prueba:", resultado)
```