Evaluación de redes neuronales artificiales tipo recurrente en el diagnóstico de fallas para el proceso Tennessee-Eastman

Tania Valentina Tiriat Villalba y Andrés Arenas Santos

Trabajo de grado para optar por el título de Ingeniero Químico

Modalidad investigación

Director

Giovanni Morales Medina

Doctor en Ingeniería Química

Universidad Industrial de Santander

Facultad de Ingenierías Fisicoquímicas

Escuela de Ingeniería Química

Bucaramanga

2024

Tabla de contenido

Lista de tablas

Lista de figuras

Lista de apéndices

Resumen

Título: Evaluación de redes neuronales artificiales tipo recurrente en el diagnóstico de fallas para el proceso Tennessee Eastman*

Autores: Tania Valentina Tiriat Villalba, Andrés Arenas Santos**

Palabras clave: Redes neuronales, Proceso Tennessee Eastman, fallas operacionales, Análisis de componentes principales, Redes recurrentes, LSTM.

Descripción: El objetivo del proyecto fue la detección de fallas en procesos químicos, utilizando la simulación Tennessee Eastman, mediante redes neuronales recurrentes con celdas LSTM programadas a través de Python. Inicialmente, se seleccionaron las fallas críticas 6 y 18. El procesamiento de estos datos se realizó mediante el análisis de componentes principales (PCA) con el fin de reducir dimensionalidad, observando al mismo tiempo la poca efectividad de este método en la detección. Se evaluaron, además, 45 modelos de LSTM con diversas configuraciones de neuronas, funciones de activación y *epochs*, logrando una precisión mínima de 76.82% y máxima del 87.62%. Las pérdidas oscilaron entre 0.063 y 0.138. También se confirmó que el aumento de los *epochs* no mejora los resultados y, al mismo tiempo, que 16 neuronas representan la configuración óptima para la detección de estas fallas, obteniéndose los mejores resultados con la arquitectura de 16 neuronas y 10 *epochs*, utilizando la función de activación Tanh. Aunque el uso del PCA como pretratamiento permitió reducir la dimensionalidad, también disminuyó la precisión en comparación con estudios anteriores, por esta razón, al evaluarlo en las otras fallas del TEP se obtuvieron resultados deficientes.

---

*Trabajo de grado

**Facultad de Ingenierías Fisicoquímicas. Escuela de Ingeniería Química. Director: Giovanni Morales Medina. Doctor en Ingeniería Química.

Abstract

Title: Evaluation of recurrent type artificial neural networks in fault diagnosis for the Tennessee Eastman process*

Authors: Tania Valentina Tiriat Villalba, Andrés Arenas Santos**

Keywords: Neural networks, Tennessee Eastman process, operational failures, Principal Component Analysis, Recurrent networks, LSTM.

Description: The objective of the project was to detect faults within chemical process by utilizing the Tennessee Eastman Process (TEP) simulation, which, through recurring neural networks with long short-term memory (LSTM) cells programmed through Python. Initially, the critical faults 6 and 18 were selected. The pre-processing of these data was performed by means of principal component analysis (PCA) in order to reduce dimensionality, a low effectiveness of PCA was also demonstrated when used as a detection method using the Hotelling statistic. Later, forty-five LSTM models with various configurations of neurons, activation functions and epochs were evaluated, achieving a minimum accuracy of 76.82% and a maximum of 87.62%. Losses ranged between 0.063 and 0.138. It was confirmed that increasing the epochs does not improve the results and at the same time that 16 neurons represent the optimal configuration for the detection of these faults, with the best results being obtained with the 16-neuron and 10-epoch architecture, using the Tanh activation function. Although the use of PCA as a pre-treatment allowed reducing the dimensionality, it also decreased the accuracy compared to previous studies, for this reason, when evaluating it on the other TEP faults, poor results were obtained.

---

*Bachelor thesis

**Faculty of Physicochemical Engineering. Chemical Engineering School. Director: Giovanni Morales Medina, Ph.D. Chemical Engineering

Introducción

En la actualidad, se presenta un constante desarrollo de las industrias además de un aumento en la demanda de productos que cumplan con determinados estándares de calidad. Lo anterior, sumado a las estrictas leyes de regulación ambiental y a la necesidad de mantener la seguridad de procesos, hace que las industrias estén en continua búsqueda de procedimientos encaminados a la protección de sus instalaciones, activos de proceso, personal e inversión (González, 2020).

En Colombia, el Decreto 1072 de 2015 adopta el programa de prevención de accidentes mayores (PPAM), estableciendo acciones, procedimientos e intervenciones que se realizan con la finalidad de mejorar los niveles de seguridad para la población y el medio ambiente; este Decreto define elementos de cumplimiento como el establecimiento de la investigación de accidentes e incidentes, además, de la inspección, vigilancia y control de estos.

Una estrategia de prevención, en línea con el Decreto 1072, corresponde a la detección y al diagnóstico temprano de fallas en los activos de procesos fisicoquímicos, por medio de la aplicación de redes neuronales artificiales (RNA). Estas RNA son algoritmos de inteligencia artificial inspirados en el funcionamiento de las neuronas y del sistema nervioso de los seres vivos (Salas, 2004). Las RNA, pueden ser entrenadas para analizar y reproducir tendencias contenidas en datos históricos de proceso. Los históricos contienen información sobre la operación diaria de una planta de procesos en un periodo o ventana de tiempo, por lo cual, estos datos establecen las tendencias de operación normal que pueden ser utilizadas en la detección de condiciones anómalas conducentes a fallas (Medrano, 2019; Cárdenas & Reyes, 2021).

Una de las RNA que pueden ser aplicadas en la detección y el diagnóstico de fallas corresponde a las redes neuronales tipo recurrente (RNR). Estas RNR son particularmente de

utilidad en el procesamiento de datos secuenciales que utilizan conexiones de retroalimentación con la finalidad de almacenar información a lo largo del tiempo (Cruz *et al*., 2007). Las RNR han sido aplicadas en problemas, tales como reconocimiento de voz, tratamiento de imágenes, modelado de lenguaje, traducción de textos (Curiel, 2022) y detección de fallas en procesos químicos (Ayodeji *et al*., 2018; Mirzaei *et al*., 2022). La aplicación de las RNR puede ser realizada mediante los códigos de uso libre del programa Python. De hecho, la librería Keras de Tensorflow, ha sido reportada en diferentes manuscritos (Cárdenas & Reyes, 2021; Curiel, 2022; Kang, 2020)

Los datos de operación de una planta real pueden ser de difícil consecución si se trata para fines de investigación académica (Arunthavanathan *et al*., 2020; Yin *et al*., 2012; Melo *et al*., 2024). Alternativamente, las investigaciones de detección y diagnóstico de fallas pueden ser adelantadas, utilizando los datos operacionales proveídos por el *Tennessee Eastman Process* (TEP), una simulación de acceso público que reporta información histórica de las variables de proceso de una planta química hipotética (Cárdenas & Reyes, 2021; González, 2020; Sheng, 2020). Los datos de simulación del TEP pueden ser aplicados en el análisis del desempeño de las RNR en la detección y el diagnóstico de fallas.

Con lo anterior, el presente documento expone los principales resultados de una investigación dirigida al análisis del desempeño de las RNR aplicadas en la detección y el diagnóstico de dos fallas codificadas en el TEP. La pregunta de investigación planteada fue, ¿Cuál arquitectura de RNR, según la librería Keras del lenguaje Python, presenta mejores desempeños en la detección y el diagnóstico de dos fallas contenidas en la simulación del TEP? En la solución de esta pregunta, la investigación fue estructurada en 7 capítulos, iniciando con la definición de los objetivos en el Capítulo 1, seguido de los fundamentos teóricos en el Capítulo 2, en donde se revisan aquellos conceptos necesarios para el desarrollo de la investigación como RNA, RNR y

LSTM. El Capítulo 3 presenta algunos reportes previos relevantes, en donde se estudia el avance en la detección y el diagnóstico de fallas en sistemas complejos mediante el uso de redes neuronales y PCA, se destaca la aplicación de técnicas recurrentes como LSTM y GRU para mejorar la precisión y rapidez en la identificación de anomalías, estos análisis revelan la efectividad de estos métodos en entornos como plantas químicas y centrales nucleares. El Capítulo 4 explica la metodología utilizada en la detección de fallas con las redes RNR y los datos de la simulación TEP, con la fase 1 de pretratamiento de datos y análisis inicial, fase 2 de modelo de redes recurrentes y fase 3 de evaluación y comparación de resultados. El capítulo final presenta los resultados más relevantes de la experimentación computacional conducente a la detección de fallas con RNR, obteniendo que el PCA no resultó eficaz en la detección de las fallas estudiadas, pero facilitó la reducción de la dimensionalidad de los datos y que en la evaluación de 45 modelos se lograron precisiones entre 76,82% y 87,62%, donde el modelo más eficiente es el que tiene una configuración de 16 neuronas, 10 *epochs* y función de activación Tanh.

Este trabajo fue financiado por el proyecto interno No 3777 “Redes neuronales artificiales en la detección y el diagnóstico de fallas: evaluación del desempeño en el proceso Tennessee Eastman”.

1. Objetivos

1.1 Objetivo general

Evaluar la eficiencia de las arquitecturas de redes neuronales recurrentes, según los algoritmos disponibles en Python, en la detección de dos fallas programadas del proceso Tennessee-Eastman.

1.2 Objetivos específicos

- Aplicar pretratamientos de datos a dos fallas del proceso Tennessee-Eastman, por medio de procedimientos estadísticos, definiendo las entradas de predicción para las RNR.
- Codificar la aplicación del entrenamiento y la validación de las RNR, considerando las rutinas disponibles en Python y los datos pretratados del proceso TEP, estableciendo los parámetros de ajuste disponibles para la detección de fallas.
- Definir los errores en la predicción de las dos fallas del proceso TEP, con base en los resultados de la aplicación del código desarrollado, determinando la estructura RNR con mejor desempeño de predicción.

2. Marco conceptual

2.1 Red Neuronal Artificial

Las Redes Neuronales Artificiales se basan en la estructura del sistema nervioso humano, el cual, está compuesto de redes de neuronas con una capacidad total de procesamiento definida por la interacción entre estas redes. La filosofía de las RNA está definida como un algoritmo o método de resolución de problemas de caja negra que puede ser aplicado en la realización de tareas como identificación, clasificación, diagnóstico, optimización o predicción a partir de un conjunto de datos iniciales. Procedimentalmente una RNA es un procesador que inicialmente suma las

entradas, luego, aplica una función que determina la activación o no de la neurona para enviar una señal a la siguiente red. La fuerza en que estas señales son ajustadas y transmitidas depende de unos factores conocidos como pesos, los cuales son esenciales para abordar los problemas utilizando un conjunto de datos representativos. Para optimizar su funcionamiento y evitar un sobreajuste en la red sobre los datos de entrenamiento, se emplean algoritmos de aprendizaje que ajustan la estructura de la red y sus parámetros internos. (Salas, 2004)

2.2 Red Neuronal Recurrente (RNR)

Se trata de un tipo de RNA que utiliza series temporales, son también conocidas como espaciotemporales o dinámicas, establece una correspondencia entre secuencias de entrada y de salida; cumple con tres tareas esenciales: reconocimiento, reproducción y asociación temporal. (Cruz *et al*., 2007)

La diferencia de este tipo de RNA radica en la utilización de la memoria adicional, es decir, se trata de una red con retroalimentación "*feedback*" que al igual que las RNA convencionales suministran una respuesta a partir de las entradas, las RNR almacenan la información de entrada en el estado de memoria, por lo que las predicciones no dependen solo de la entrada sino también de los datos anteriores. (Kostadinov, 2018)

Para un correcto funcionamiento de las redes es importante realizar el proceso de establecer la configuración óptima de hiperparámetros, los cuales son variables ajustables de la red que adaptan el entrenamiento de un sistema y de estos depende en gran parte el rendimiento del modelo (Amibp, 2023), los más utilizados a lo largo del documento serán descritos a continuación:

- El *batch size* se trata de la cantidad, tamaño o número de ejemplos que se introducen en la red cada vez que se entrena, si este parámetro es pequeño quiere decir que hay poca cantidad de

datos dentro de la memoria y por ende se entrena más rápido, pero puede darse el caso de que no obtenga y aprenda las características significativas para la predicción. (Casas, 2020)

- Un *epoch* representa un ciclo completo en el conjunto de entrenamiento, este está compuesto por lotes *"batch"* y múltiples iteraciones, en otras palabras, se trata de las veces que se va a introducir cada ejemplo en la red; el número requerido de esta variable para un proceso eficiente depende de la naturaleza de los datos y el objetivo del modelo, normalmente es necesario utilizar varios *epochs* con el fin de alcanzar el resultado deseado en el aprendizaje automático. (Ciberseg, 2021)

- La función de activación es aquella que se desempeña como filtro o umbral y modifica el resultado imponiendo un límite máximo que no puede sobrepasarse si es que se quiere pasar a la siguiente neurona, es decir, transfiere la información que se genera a partir de la combinación lineal de los pesos y las entradas, por tanto, es el puente que transmite la información a las conexiones de salida; dentro de las funciones más conocidas se encuentran la sigmoidal, la tangente hiperbólica y la exponencial normalizada. (Tech, s. f.)

Existen distintos tipos de redes neuronales recurrentes, tales como: RNR estándar, RNR de memoria de corto-largo plazo (*Long short term memory*, LSTM), RNR de puertas (*General recurrent unit*, GRU), LSTM bidireccional, entre otras. (Salem, 2022)

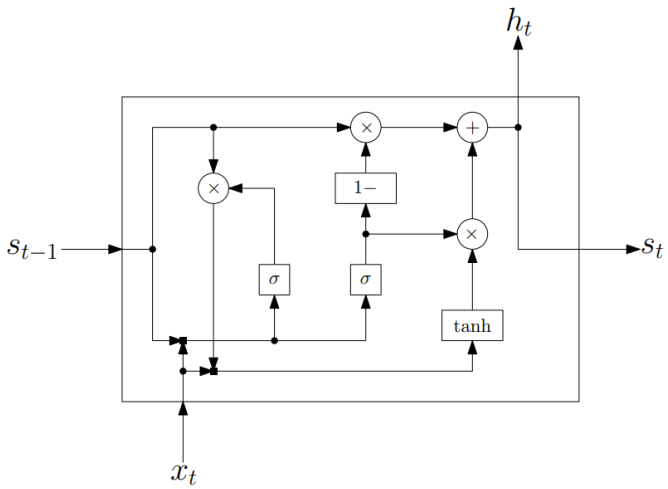
2.2.1 *Red neuronal de memoria de corto-largo plazo (LSTM)*

Esta RNR de análisis de datos, conocida en inglés como *Long short term memory* (LSTM), puede aprender relaciones complejas entre puntos temporales secuenciales, permitiendo capturar dependencias tanto entre puntos lejanos como cercanos. Para que la LSTM adquiera esta habilidad es necesario que contenga dos puntos claves, canal de memoria y control de flujo de información

que se realiza mediante compuertas, en la Figura 1 se puede evidenciar la arquitectura de una neurona o celda de este tipo de red neuronal recurrente. Esta se basa en controlar cuál es la información que se guarda en el estado, fluyendo por la parte superior de la celda y que se produce como respuesta en la red, este control se ofrece sobre tres compuertas: olvido, entrada y salida. (Redes Neuronales de Memoria de Corto-largo Plazo - MATLAB & Simulink - MathWorks América Latina, s. f.)

Figura 1

Diagrama de una celda LSTM



Nota. Adaptado de (Rué *et al.*, 2019)

Con el fin de medir el rendimiento de las RNR se utilizan dos parámetros, precisión y error cuadrático medio (*Mean squared error*, MSE) los cuales son descritos a continuación por las ecuaciones 1 (Mirzaei *et al*., 2022) y 2 respectivamente, donde N representa el número total de conjunto de datos, y_i es el valor objetivo real i, $\hat{y}_i$ es el valor estimado correspondiente i (Sheta *et al*., 2009).

$$Precisión = \frac{Número\ de\ muestras\ correctamente\ categorizadas}{Número\ de\ muestras\ totales} \quad (Ec.\ 1)$$

$$MSE = \frac{1}{N}\sum_{i=1}^{i=N}(y_i - \hat{y}_i)^2 \quad (Ec.\ 2)$$

2.3 Proceso Tennessee Eastman (TEP)

Se trata de un problema de control propuesto en 1993 por ingenieros de la empresa *Eastman Chemical*, los cuales crearon una planta química hipotética con el fin de estudiar métodos de control de procesos y extrapolarlos a otros procesos u otras plantas (Downs & Vogel, 1993). El proceso contiene ocho componentes, que fueron nombrados de la A a la H, cuatro son reactivos, dos productos, un inerte y una purga; ocurren cuatro reacciones todas reversibles y exotérmicas, las ecuaciones 3 y 4 representan las dos reacciones principales y las ecuaciones 5 y 6 las secundarias, es decir G y H son considerados productos y F es un subproducto.

$$A_{gas} + C_{gas} + D_{gas} \rightarrow G_{líquido} \quad (Ec.\ 3)$$

$$A_{gas} + C_{gas} + E_{gas} \rightarrow H_{líquido} \quad (Ec.\ 4)$$

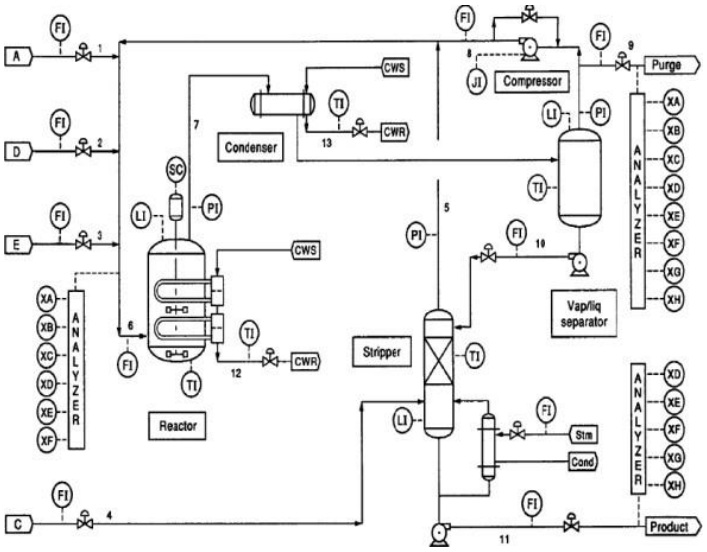
$$A_{gas} + E_{gas} \rightarrow F_{líquido} \quad (Ec.\ 5)$$

$$3D_{gas} \rightarrow 2F_{líquido} \quad (Ec.\ 6)$$

Además, contiene cinco equipos clave: reactor, condensador, compresor, separador y columna destiladora como se puede ver en la Figura 2. El TEP posee 41 instrumentos de medición y 12 variables que pueden manipularse, descritas en las Tablas 1 y 2 respectivamente, en estas se puede ver cómo está codificado y qué representa cada una de las variables descritas en el proceso.

Figura 2

Diagrama de flujo de proceso Tennessee Eastman



Nota. Adaptado de (Downs & Vogel, 1993b)

La simulación consiste en 22 ensayos también llamados grupos de datos, de los cuales 21 son para operaciones de la planta con distintas fallas y el restante para una operación normal, se puede encontrar la descripción de estos ensayos en el Apéndice A (Chiang *et al*., 2001); las fallas 6 y 18 son la estudiadas a lo largo del documento y están representadas como IDV (6) e IDV (18) dentro de los paquetes de datos descargados, en el Apéndice A se encuentra la descripción más detallada de los nombres, su representación en el proceso y de qué tipo de falla se trata.

Tabla 1

Variables medidas del TEP

Variable	Nombre de la variable	Unidades
XMEAS (1)	Flujo de alimentación de A	kscmh
XMEAS (2)	Flujo de alimentación de D	kscmh
XMEAS (3)	Flujo de alimentación de E	kscmh
XMEAS (4)	Flujo de alimentación de A Y C	kscmh
XMEAS (5)	Flujo de recirculación	kscmh
XMEAS (6)	Flujo de alimentación al reactor	kscmh
XMEAS (7)	Presión del reactor	kPa
XMEAS (8)	Nivel del reactor	%
XMEAS (9)	Temperatura del reactor	°C
XMEAS (10)	Flujo de purga	kscmh
XMEAS (11)	Temperatura del separador	°C
XMEAS (12)	Nivel del separador	%
XMEAS (13)	Presión del separador	kPa
XMEAS (14)	Corriente del separador	m^3/h
XMEAS (15)	Nivel de destilador (stripper)	°C
XMEAS (16)	Presión del destilador (stripper)	kPa
XMEAS (17)	Corriente del destiladort (stripper)	m^3/h
XMEAS (18)	Temperatura del destilador (stripper)	°C
XMEAS (19)	Flujo de vapor del destilador (stripper)	kg/h
XMEAS (20)	Potencia de compresor	kW
XMEAS (21)	Temperatura de la salida del agua de refrigeración del reactor	°C
XMEAS (22)	Temperatura de la salida del agua de refrigeración del separador	°C
XMEAS (23-28)	Concentración de alimentación del reactor (A-F)	%mol
XMEAS (29-36)	Concentración de la purga (A-H)	%mol
XMEAS (37-41)	Concentración aguas abajo del destilador (A-H)	%mol

Nota. Adaptado de (Downs & Vogel, 1993b)

Tabla 2

Variables manipulables del TEP

Variable	Nombre de la variable	Unidades
XMV (1)	Flujo de alimentación D	kg/h
XMV (2)	Flujo de alimentación E	kg/h
XMV (3)	Flujo de alimentación A	kscmh
XMV (4)	Flujo de alimentación A Y C	kscmh
XMV (5)	Válvula de recirculación compresor	%
XMV (6)	Válvula de carga	%
XMV (7)	Flujo de líquido del separador LV	m^3/h
XMV (8)	Flujo de vapor de la columna de stripping	m^3/h
XMV (9)	Válvula de vapor de la columna de stripping	%
XMV (10)	Flujo de agua de refrigeración del reactor	m^3/h
XMV (11)	Flujo de agua en el condensador	m^3/h

Nota. Adaptado de (Downs & Vogel, 1993b)

Agregando a lo anterior, los autores del proceso Tennessee Eastman asignaron los límites de operación normal y los que generarían un paro del sistema para cinco variables que corresponden a tres equipos (reactor, separador y columna de destilación). Esta información se puede ver en la Tabla 3.

Tabla 3

Límites de variables correspondientes a reactor, separador y columna de destilación

Variable	Límites de operación normales		Límites que provocan paro del sistema	
	Limite inferior	Límite superior	Limite inferior	Límite superior
Presión del reactor	Ninguno	2896 kPa	Ninguno	3000 kPa
Nivel del reactor	50% (10.65 m^3)	100% (21.3 m^3)	2.0 m^3	24 m^3
Temperatura del reactor	Ninguno	150°C	Ninguno	175°C
Nivel del separador	30% (3.0 m^3)	100% (9 m^3)	1.0 m^3	12.0 m^3
Nivel de columna	30% (1.98 m^3)	100% (6.6 m^3)	1.0 m^3	8.0 m^3

Nota. Adaptado de (Downs & Vogel, 1993b)

2.4 Análisis por componentes principales (PCA)

Es una técnica descriptiva utilizada para analizar datos cuantitativos en múltiples dimensiones, reduciendo la dimensionalidad original mediante la proyección de los puntos sobre un espacio de dimensión menor manteniendo las posiciones relativas y conservando la variabilidad de los puntos, permitiendo al mismo tiempo la detección de fallas en primera instancia. En el contexto de los procesos industriales, se tienen gran cantidad de variables interrelacionadas, esto se conoce como alta dimensionalidad con una colinealidad significativa, lo que conduce a la presencia de ruido en los datos y motiva a utilizar el control estadístico de procesos multivariante. (Rué *et al.*, 2019)

Para comenzar es de importancia normalizar las variables, antes de aplicar esta técnica, con el fin de que todas tengan el mismo peso, el procedimiento que se realiza para lograr esto, conocido como normalización por autoescalado, es el que se muestra en la ecuación 7.

Por otro parte, el PCA también permite la detección de fallas mediante el estadístico Hotelling (T^2) y su desempeño puede ser medido por la tasa de detección de pérdidas (*Missed detection rate*, MDR) que se define en la ecuación 8. (Sheng, 2020)

$$x_{normalizado} = \frac{x-\mu}{\sigma} \quad (Ec.\ 7)$$

$$MDR = \frac{número\ de\ fallas\ no\ detectadas}{número\ total\ de\ fallas} \quad (Ec.\ 8)$$

3. Estado del arte

Ayodeji *et al*. (2018), aplicaron análisis de componentes principales (PCA) y dos RNR, la red neuronal Elman (ENN) y la red de base radial (RBFN) en la predicción y el diagnóstico de fallas en la simulación de una central nuclear para entrenamiento de operadores. Según los autores, la RNR de base radial reportó una mejor predicción y diagnóstico de fallas en los diferentes escenarios analizados, con una estructura con 300 *epochs* y 311 neuronas en la capa oculta, obteniendo un tiempo de entrenamiento 1.27 segundos, una precisión promedio de regresión en los tres conjuntos de datos de 0.974 y un MSE de 0.04.

Medrano (2019), estudió técnicas de clasificación de diagnóstico de fallos como lo son las redes neuronales de perceptrón multicapa (MLP), redes probabilísticas (PN), máquinas de vectores soporte (SVM), árboles de decisión y discriminante de Fisher utilizando los datos de los fallos 1, 2, 4, y 5 del conocido proceso Tennessee Eastman. Concluyó que se debe descartar las PN como

clasificadoras y se obtuvieron los mejores resultados con los árboles de decisión con porcentajes de clasificación cercanos al 100% con ventajas como programación sencilla y tiempo de clasificación corto. Plantea para trabajos futuros realizar un ensamble de los clasificadores utilizados exceptuando el PN, utilizando técnicas de ensamble como *Bagging*, *Boosting* y *Stacking*; también menciona la opción de utilizar el análisis de componentes principales (PCA) como un paso previo a la utilización de las técnicas de clasificación.

Kang (2020), evaluó el diagnóstico de fallas de detección completa y temprana en procesos químicos complicados utilizando RNR en diferentes configuraciones, variando capas y nodos. Finalmente comparó los resultados obtenidos con RNA, demostrando que las recurrentes tienen mejores calificaciones de precisión sin importar la calidad del conjunto de datos utilizados en el entrenamiento. Estos resultados se obtuvieron gracias a que las RNR sacan los datos operativos normales del centro y crean una distribución espacial uniforme de los tipos de fallas, dentro de los resultados se encuentra que la mejor precisión promedio fue del 84.9% para RNR con una capa y 256 nodos partiendo del conjunto de datos ajustado, pese a que se obtuvieron buenos resultados no se logró predecir la falla 15, por lo que se optó por realizar una mejora con la adición de la celda LSTM con una sola capa y 16 nodos, resultando en que esta puede predecir todas las fallas incluyendo la 15, pero el autor no hizo énfasis en este descubrimiento y lo propuso como una mejora al proceso.

González (2020), desarrolló técnicas de detección y diagnóstico de fallas basadas en datos del TEP, para la parte del análisis estadístico del proceso se utilizó PCA y para el diagnóstico y clasificación se usaron RNA y LSTM con el fin de realizar una comparación de las dos en las mismas circunstancias computacionales. Se obtuvieron resultados para la RNA con un aumento en la eficacia cuando se tienen tres capas en vez de una sola y también se obtiene una mejoría si

se trabaja con un número de neuronas de 100 en vez de 30 o 50 con una eficacia del 73.4%. Por otro lado utilizando PCA en este mismo tipo de red no se obtiene un aumento significativo de la eficacia de clasificación pero si se tiene más tiempo de entrenamiento y coste computacional; para el caso de las LSTM no se han ofrecido resultados concluyentes referentes a la clasificación a partir de los datos de partida, arrojando que es necesario utilizar datos mucho más extensos para el entrenamiento alcanzándose un 99% de eficiencia cuando se amplía la matriz, el autor sugiere seguir trabajando en redes LSTM utilizando diferentes combinaciones que mejoren la eficacia.

Pequeño (2020), utilizó el PCA fuera de línea como técnica estadística de control de procesos, sobre la planta hipotética Tennessee Eastman, posteriormente hizo uso de redes de propagación hacía adelante que no arrojaron buenos resultados en ningún caso, teniendo un mejor funcionamiento las redes de aprendizaje profundo con dos capas ocultas y 1 capa de clasificación de propagación hacía adelante. Dadas las pruebas de precisión de regresión los resultados fueron parecidos entre el RBFN y el ENN, sin embargo, en el RBFN con 311 neuronas en la capa oculta demostró ser más rápido en el entrenamiento que ENN; una técnica propuesta por el autor como una mejora para clasificar fallos, fueron los bosques aleatorios *"Random Forest",* esta demuestra ser útil en la búsqueda de anomalías y la selección de predicciones.

Cárdenas y Reyes (2021), Determinaron la estructura de una RNA óptima para detectar fallas operacionales a partir del TEP, utilizando el software R con el paquete Neuralnet y Python utilizando la interfaz Keras. Plantearon el código considerando fallas y operación estable del proceso; variando parámetros como la cantidad de neuronas, función de activación y número de capas ocultas, se obtuvo que la red que mejor se adapta es la que tiene una estructura 52:11:1 la que presenta un menor error cuadrático medio de 0.01823 y una eficiencia de predicción del 97% en la detección de fallas. Recomiendan el uso de la herramienta PCA, utilizar más datos y

parámetros más exigentes en el software R, para que se vea reflejada en la eficiencia y velocidad de aprendizaje de RNA.

Curiel (2022), Utilizó PCA y autoencoders simples, recurrentes, GRU, variacionales y variacionales recurrentes con el fin de encontrar cual es la mejor técnica para detectar fallos en la planta química Tennessee Eastman. Consiguieron detectar 17 de los 21 fallos posibles de la planta y se obtuvieron los mejores resultados con una tasa de detección media del 81.09% con el Autoencoder variacional decreciente-creciente. Los autores recomiendan utilizar en trabajos futuros la red LSTM y la red recurrente GRU para conseguir mejores resultados combinando distintas funciones de activación tanto en la función de propagación hacia adelante como en la función recurrente.

Mirzaei *et al*. (2022), Utilizaron la RNR con unidades cerradas LSTM y GRU para proporcionar información comparativa sobre los modelos adecuados en el diagnóstico de fallas en procesos químicos, utilizando los datos del TEP; como resultados se obtuvo que el modelo GRU mostró superioridad y separó de mejor manera las fallas, en especial la 15, LSTM presentó una precisión del 63% y GRU aumentó en un 13% su precisión hasta llegar al 76%.

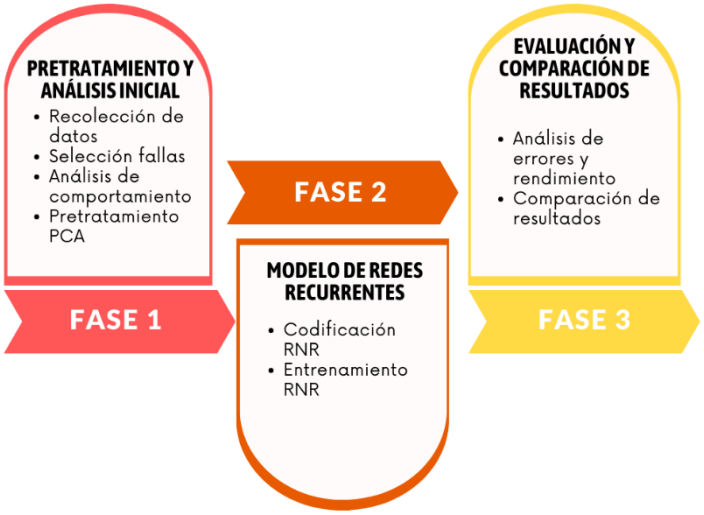
Tras revisar la serie de investigaciones y trabajos académicos presentados arriba, relacionados con el proceso químico TEP y la detección y diagnóstico de fallos, es posible afirmar que, la RNR representa una mejora con respecto al rendimiento de una RNA convencional. Asimismo, las RNR más reportadas en los estudios son las de tipo LSTM y GRU. Los estudios sugieren que la aplicación previa de pretratamientos a los datos como el PCA conducen a una mejora significativa en las predicciones. Con lo anterior, se observa un potencial de aplicación de las RNR en lo que respecta a la detección temprana de anomalías en entornos industriales.

4. Metodología

La metodología seguida en la presente investigación es ilustrada en la Figura 3. Las fases en las que fue dividida la metodología, correspondientes con los objetivos específicos, son detalladas a continuación.

Figura 3

Diagrama de flujo metodológico



Fase 1: Pretratamiento de datos y análisis inicial

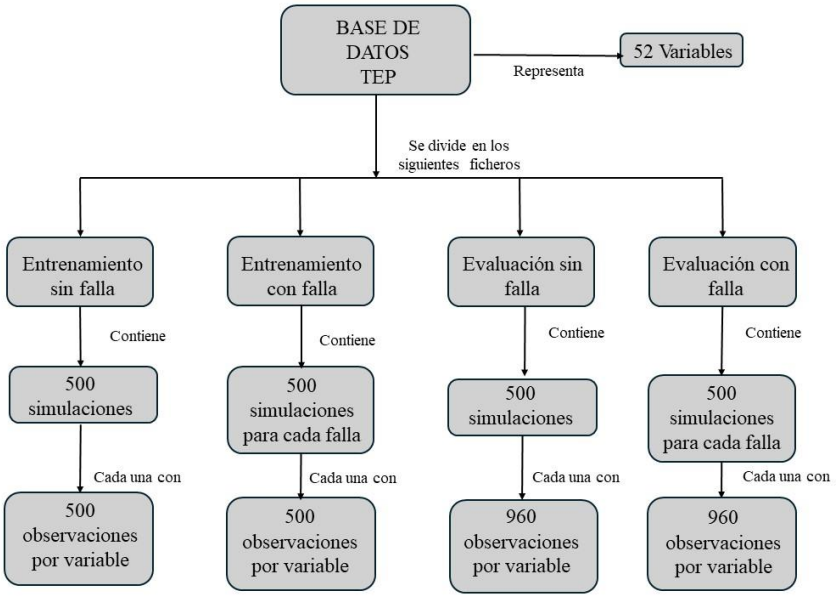
Actividad 1. Recolección de datos donde se descargaron conjuntos de información adecuada del TEP.

La base de datos del proceso Tennessee Eastman es de acceso público y está disponible en varias fuentes; para este trabajo se obtuvo del servidor de la universidad de Harvard (Rieth *et al*., 2017). El paquete obtenido allí está conformado por cuatro ficheros con observaciones tomadas cada 3 minutos de las 52 variables medidas y manipuladas del proceso. Dos de estos ficheros corresponden a los datos para el entrenamiento "*training*" del modelo de detección de fallas y los

otros dos para realizar la evaluación "*testing*" del mismo. La Figura 4 representa mediante un esquema la estructura de esta base de datos.

Figura 4

Esquema explicativo de la base de datos del TEP



Los dos archivos de entrenamiento incluyen 500 simulaciones para cada uno de los estados posibles (sin falla y con cada una de las fallas), cada simulación equivale a una corrida del proceso, y cada una de estas contiene 500 observaciones, lo que corresponde a 25 horas de registro de operación. Uno de los archivos registra el funcionamiento normal de la planta y el otro documenta las primeras 20 fallas, ya que la base de datos no incluye información de la falla 21, las fallas se encuentran listadas en el Apéndice A. La base de datos establece 500 simulaciones por cada falla, con la aparición de la respectiva falla desde el inicio de cada simulación. Mientras que, los dos archivos destinados para la evaluación contienen igual número de simulaciones por estado, pero con 960 observaciones cada una, lo que equivale a 48 horas de operación de la planta. En este caso

la información también viene segregada por datos sin falla y con fallas en distintos archivos, en el caso de las fallas estas fueron introducidas en cada simulación después de 8 horas de operación.

Actividad 2. Definición y selección de las dos fallas de mayor impacto en el proceso Tennessee Eastman.

En la presente investigación, se seleccionaron las fallas 6 y 18 para realizar el análisis, ya que estas fallas son consideradas las más críticas del proceso (Hajihosseini *et al*., 2018), la aparición de estas fallas deriva en parada de planta, por lo cual, es necesaria su detección temprana (Lyman & Georgakis, 1995). La falla 6 de tipo escalón, implica la pérdida de alimentación del componente A, la ausencia de este reactivo detiene la reacción química. En la aparición de esta falla, los niveles de los otros reactivos (D y E) aumentan, conduciendo a un incremento en la presión en el reactor hasta llegar al nivel del límite de seguridad (Chiang *et al*., 2001a). Por otro lado, la falla 18 se considera como una de las desconocidas.

De los cuatro ficheros proveídos por la base de datos se extrajeron las primeras 50 simulaciones tanto de los datos sin falla como de los datos con las dos perturbaciones elegidas. De esta manera, se obtuvieron tres grupos de 50 matrices, cada una con 500 filas (muestras) y 52 columnas (variables) para el entrenamiento del modelo de detección, así mismo se prepararon 3 conjuntos con matrices de tamaño (960x52) para la última fase del estudio. Esta selección no incluyó la totalidad de las 500 simulaciones disponibles con el fin de reducir los tiempos de computación del entrenamiento de la red neuronal y evitar un sobredimensionamiento de esta, una estrategia que también usó Kang (2020) en su estudio.

Actividad 3. Análisis de comportamiento de datos para las fallas 6 y 18 identificando patrones y características relevantes tales como impacto de variables de perturbación y de salida además de las zonas de estado pseudoestacionario.

El proceso se realizó con los datos de la primera simulación de entrenamiento de la operación sin falla unida a la primera simulación de entrenamiento de cada una de las fallas seleccionadas, generando así dos nuevas simulaciones de 1000 observaciones en total, que permiten visualizar de forma clara el estado de la operación normal como de las perturbaciones. Esta técnica de enriquecimiento de datos fue empleada también por Sheng (2020), debido a la limitada cantidad de muestras para la operación normal en las matrices de entrenamiento con falla.

Posteriormente, se examinó el comportamiento de las cinco variables cuyos límites de operación normal se encuentran en la Tabla 3 (presión del reactor, temperatura del reactor, nivel del reactor, nivel del separador y nivel de la columna). Además, se analizó el comportamiento de dos variables adicionales para cada perturbación. En el caso de la falla 6, se evaluó el comportamiento del flujo de alimentación de A y la apertura de la válvula de alimentación del mismo componente, que son las dos variables relacionadas directamente con ella. (Chiang et al., 2001a) Mientras que, para la falla 18 se analizó adicionalmente el comportamiento de la presión del separador y la presión de la columna.

Actividad 4. Pretratamiento con PCA con el fin de reducir la dimensionalidad de los datos, eliminando correlaciones para lograr como resultado unos datos más limpios y procesables para servir como entrada.

El primer paso en el pretratamiento de los datos consistió en la normalización mediante el método de autoescalado (Ec. 7), con el objetivo de estandarizar las matrices de manera que todas las columnas, que representan las variables tuvieran una media de 0 y una desviación estándar de 1. Este proceso se hace para que todas las variables tengan una escala común, otorgándoles la misma relevancia en análisis posteriores. A continuación, se realizó un análisis por componentes

principales (PCA), el cual facilita la observación del comportamiento de las fallas mediante el estadístico Hotteling (T^2), utilizado como umbral para el monitoreo del estado del proceso.

Por otra parte, los resultados del PCA fueron utilizados para el análisis gráfico de la varianza representada por cada componente y la reducción en la dimensionalidad.

Fase 2: Modelo de redes neuronales recurrentes

Actividad 5. Codificación de la red neuronal desarrollada en *Google Colab* para la aplicación de funciones de Python, que conllevara a la implementación exitosa de la arquitectura de la RNR teniendo en cuenta número de neurona internas, función de activación y *epochs*.

La aplicación de la red neuronal recurrente tipo LSTM para la detección y el diagnóstico de fallas se llevó a cabo mediante la combinación de distintos parámetros. Se probaron tres funciones de activación (Softmax, Tanh, Sigmoid) en la salida de la red, junto con 3 números de *epochs* (10, 20, 30) para cantidades de neuronas (16, 32, 64, 128, 256). En total, se evaluaron 45 modelos en Python, organizados en 5 grupos de 9 modelos cada uno. Es decir, se ejecutó cada número de neuronas por separado con los diferentes valores de *epochs* y funciones de activación.

Los modelos consistían en redes de una sola capa, se decidió mantener constante este parámetro, ya que Kang (2020) demostró en su estudio que los resultados de precisión con la LSTM desmejoran al utilizar múltiples capas. Además, el parámetro *batch size* se mantuvo constante en 128, siguiendo la misma estrategia utilizada por Mirzaei *et al.* (2022) en su trabajo.

Actividad 6. Entrenamiento de la RNR con datos, otorgando la capacidad de aprender patrones y relaciones, además de variar los parámetros de entrenamiento para saber cómo afectaban estos al desempeño de la red.

Como se mencionó anteriormente, para el proceso de entrenamiento se usaron 150 matrices que incluían la información tanto de las dos fallas como de la operación normal del sistema; todas estas matrices luego de ser pretratadas en la fase anterior fueron concatenadas en la actual fase dentro de un solo arreglo matricial llamado *X_train*, agrupando así los valores de entrada. Esto permitió que la red neuronal aprendiera el comportamiento normal y con anomalías para poder detectarlo posteriormente con datos desconocidos. Por otro lado, el arreglo *Y_train* contenía las variables de salida que cada modelo debía predecir, estas variables correspondían a los tres estados posibles (sin falla, falla 6 y falla 18) que fueron etiquetadas como (0, 1 y 2) respectivamente.

Los datos de evaluación estaban compuestos también por 150 conjuntos de datos en forma de matrices. Estos conjuntos se dividieron en dos grupos: el 20% de las 50 simulaciones de cada estado se asignaron para la validación, mientras que el 80% restante se destinó para la evaluación. La información para validación se concatenó en un solo arreglo con el nombre de *X_valid*, mientras que los datos para evaluación se agruparon bajo el nombre de *X_test*. Además, los arreglos, *Y_valid* y *Y_test* contenían las mismas etiquetas de *Y_train* (0, 1 y 2). La información de entrenamiento y validación se introdujo simultáneamente en cada modelo, con el propósito de evitar el sobreajuste de la red neuronal.

Fase 3: Evaluación y comparación de resultados

Actividad 7. Análisis de la precisión y del error cuadrático medio (MSE, por sus siglas en inglés) de predicción para comprender limitaciones y áreas de mejora del modelo RNR propuesto.

En cada ejecución de código (un total de 5), se extrajo los resultados promedio de precisión y pérdidas (MSE) del entrenamiento antes de continuar con la evaluación, de la cual se tomaron las mismas dos medidas de desempeño

Actividad 8. Comparación de resultados obtenidos con el modelo RNR y los estándares definidos o con otros modelos y pretratamientos, con el fin de evaluar la eficiencia de este.

Para comparar el desempeño de los modelos, los resultados más importantes para tener en cuenta fueron la precisión y el error promedio durante la evaluación, ya que demuestran cómo se comporta la red ante unos datos nuevos y totalmente desconocidos. Como último paso en esta fase, se generaron tablas individuales con los resultados en la evaluación de cada número de neuronas y gráficas individuales para cada cantidad de *epochs* con los resultados en la evaluación también. Lo anterior se hizo con el fin de comparar la influencia del aumento de neuronas y *epochs* respectivamente en el desempeño de la red.

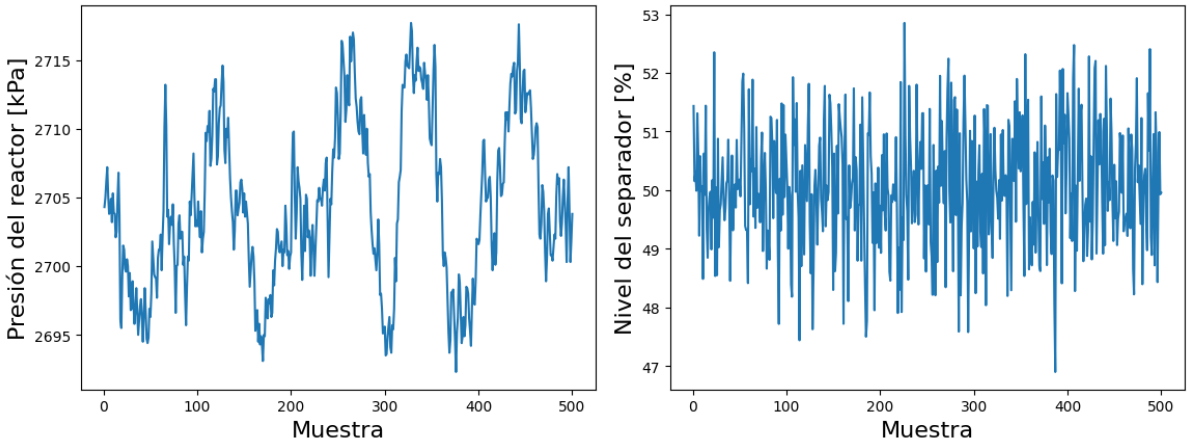
5. Resultados

5.1 Análisis de datos del proceso Tennessee Eastman.

En la Figura 5 se presenta el comportamiento de la presión del reactor y el nivel del separador, correspondientes a la primera simulación de los datos de entrenamiento sin falla. El Apéndice B contiene las gráficas del comportamiento de las variables restantes, cuyos límites de operación se detallan en la Tabla 3, en estas gráficas se observa que los valores de las cinco variables se mantienen dentro del rango establecido para una operación normal.

Figura 5

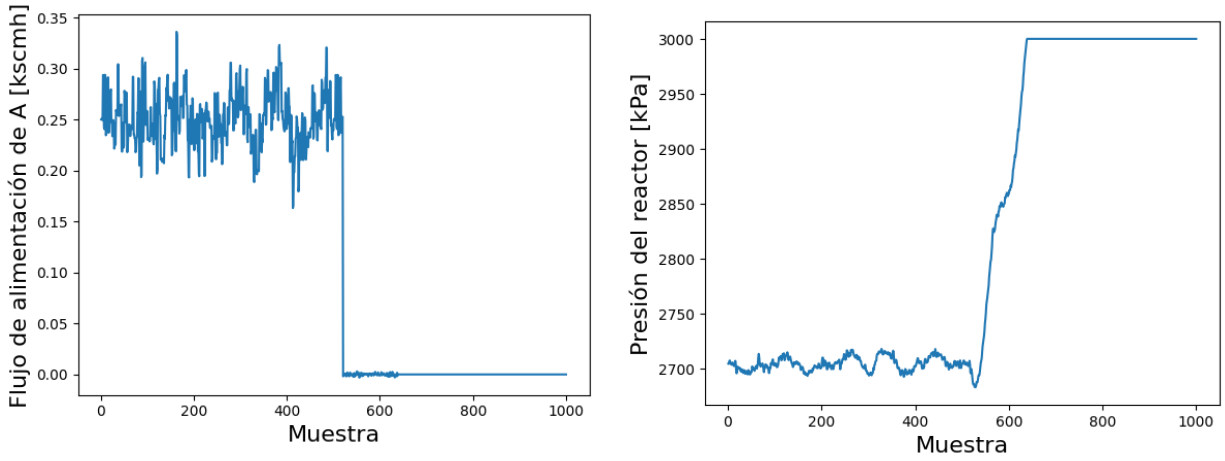
Comportamiento de la presión del reactor y el nivel del separador para la primera simulación sin falla



En la Figura 6 se incluyen las gráficas del flujo de alimentación de A y la presión del reactor para la operación con la falla 6. Se observa que la primera variable presenta un comportamiento normal hasta la muestra 520, donde tiene un descenso abrupto hasta llegar a cero, indicando el cese de la alimentación de A y causando la perturbación en el sistema. Por otro lado, la presión del reactor se mantiene aproximadamente en un valor normal de 2700 kPa hasta la muestra 538, momento en el cual la variable comienza a aumentar hasta llegar, 100 muestras más adelante, al límite de seguridad de 3000 kPa y donde se detiene la operación por precaución. El comportamiento gráfico de las restantes variables analizadas se incluye en el Apéndice D.

Figura 6

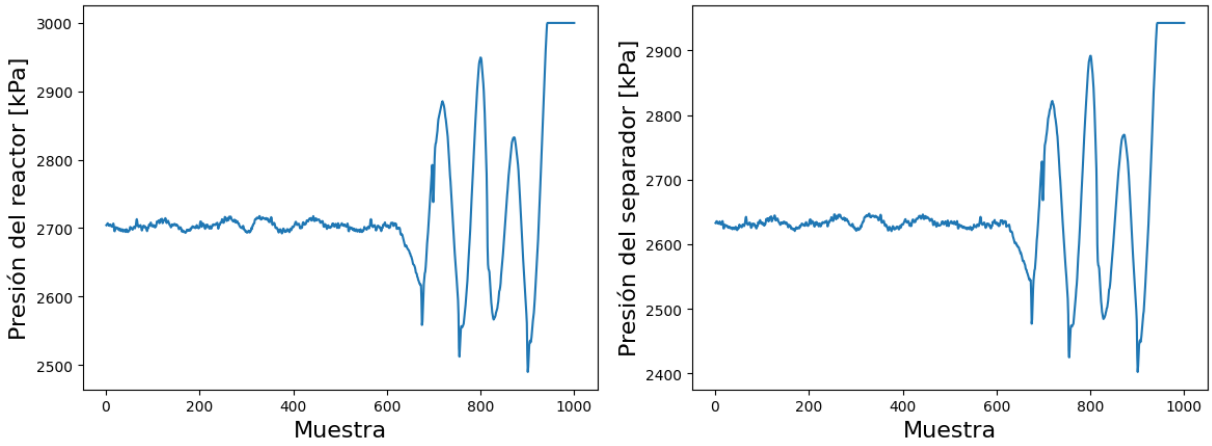
Representación gráfica del flujo de alimentación de A (izq.) y presión del reactor (der.) durante la operación con la falla 6



En la Figura 7 se encuentra el comportamiento de la presión del reactor y la presión del separador en los datos correspondientes a la falla 18. En este caso, la presión del reactor se mantiene aproximadamente en 2700 kPa hasta la muestra 600, donde su valor empieza a tener grandes fluctuaciones, ya en la muestra 800, ha superado el límite superior de operación normal (2896 kPa) y alcanza el límite de seguridad (3000 kPa) en la muestra 943, donde ocurre la parada de planta. Por otro lado, la presión del separador tiene un comportamiento gráfico similar al de la variable anterior, cambiando solo en el valor de sus datos, que llegan a un máximo de 2942.6 kPa. En el apéndice D, se puede observar el comportamiento gráfico de las demás variables analizadas para esta falla.

Figura 7

Representación gráfica de la presión del reactor (izq.) y presión del separador (der.) durante la operación con la falla 18



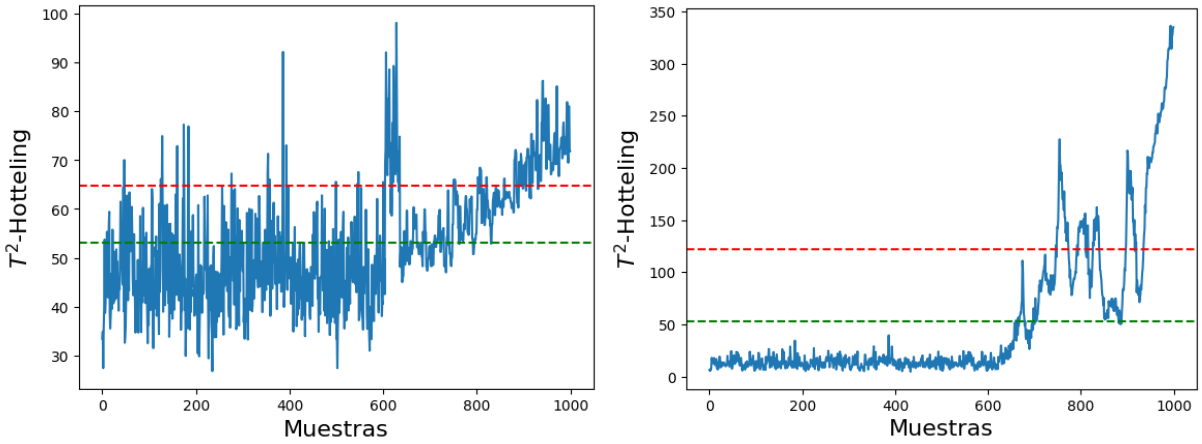
5.2 Pretratamiento de datos

Los resultados del monitoreo del proceso mediante el estadístico Hotteling (T^2) pueden apreciarse gráficamente en la Figura 8, donde junto al T^2 se aplicó un límite, representado por una línea roja, definido como la suma de la media del Hotelling (línea verde) y su desviación estándar. Ambas gráficas se asemejan a las obtenidas por Sheng (2020).

La aplicación de este método de detección de fallas reportó una tasa de detección de pérdidas (MDR) de 0.838 para la falla 6 y de 0.852 para la falla 18. En cuanto al MSE los resultados fueron de 1.912 y 2.629 para las fallas 6 y 18, respectivamente. Un valor de MDR cercano a la unidad indica que el PCA no es capaz de detectar estas fallas (Sheng, 2020). De hecho, este método estadístico asume relaciones lineales entre las variables y no es recomendado para procesos altamente no lineales como el TEP (Bishop, 2006).

Figura 8

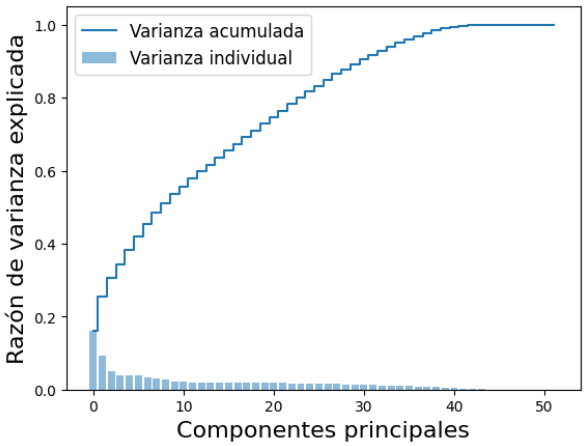
Análisis de los datos de la falla 6 (izq.) y 18 (der.) mediante el estadístico Hotteling (T^2)



En las figuras 9 y 10 se incluyen los resultados gráficos del PCA, estas muestran la varianza representada por cada componente principal para los datos de entrenamiento y en el Apéndice C se encuentran las correspondientes a las matrices de evaluación.

Figura 9

Varianza representada por los componentes principales para los datos de entrenamiento sin falla

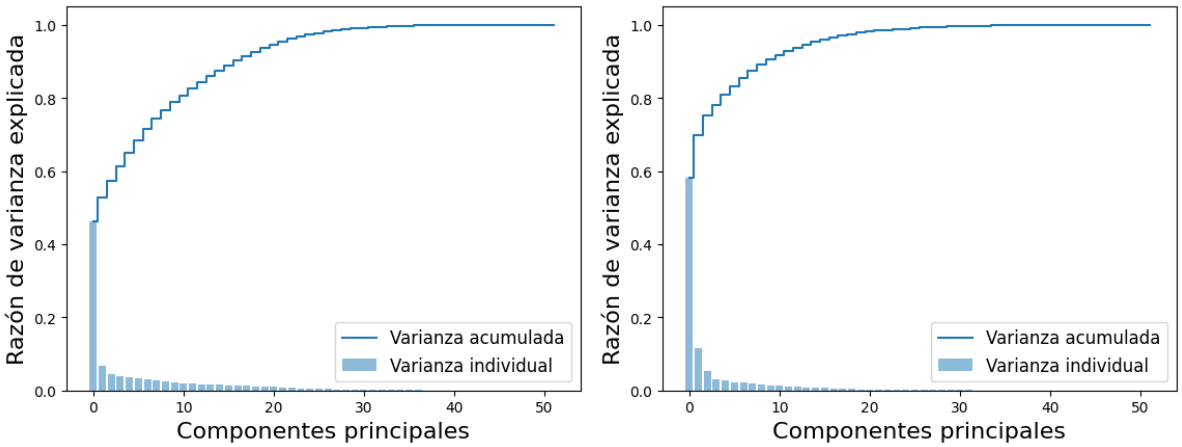


Se decidió conservar un mínimo del 90% de varianza en la reducción de componentes. Para lograr este porcentaje en los datos sin falla, se requerían 31 componentes principales, lo que coincide con el resultado del trabajo de González (2020).

En el caso de los datos de la falla 6, el 90% de la varianza se podía representar con los primeros 17 componentes principales; por otro lado, para la falla 18, este porcentaje se alcanzaba con los primeros 10. En ambos casos, el primer componente representaba un porcentaje alto de varianza, a diferencia de los datos sin falla.

Figura 10

Varianza representada por los componentes principales para los datos de entrenamiento de la falla 6 (izq.) y 18 (der.)



Aunque la cantidad de componentes necesarios para representar la varianza deseada variaba para cada estado de la operación, todas las matrices debían ser reducidas al mismo número de componentes, esto se debe a que al introducir los conjuntos de datos dentro de la red neuronal estos deben tener igual cantidad de columnas; por lo tanto, se tomó la decisión de reducirlos a 31 componentes, asegurando que todos los datos representaran al menos el 90% de la variabilidad, de

esta manera, las matrices de trabajo fueron redimensionadas a (500x31) las de entrenamiento y (960x31) las de evaluación.

5.3 Detección y diagnóstico de fallas críticas

Las figuras 11 y 12 presentan graficados los resultados de precisión y pérdidas promedio en el entrenamiento y la evaluación para las RNR tipo LSTM con 16 neuronas. En el Apéndice E se encuentran estos resultados gráficos para números superiores de neuronas, en los que se resalta que, como era de esperar, los valores siempre son mejores en el entrenamiento (mayor precisión y menores pérdidas) que en la evaluación. Además, se observa que el aumento del número de *epochs* tiende a mejorar los resultados del entrenamiento, pero no afecta positivamente a los de la evaluación que se mantienen similares o incluso desmejoran al incrementar este parámetro. Una explicación lógica para esto es que más *epochs* equivalen a más tiempo de entrenamiento y después de cierto punto es ineficiente aumentar este parámetro e incluso se puede presentar sobreajuste, en donde la red memoriza los datos de entrenamiento en lugar de generalizarlos. Lo anterior genera un buen rendimiento del modelo con los datos de entrenamiento, pero deficiente con datos nuevos.

Figura 11

Precisión promedio para las LSTM con 16 neuronas entrenamiento (izq.) y evaluación (der.)

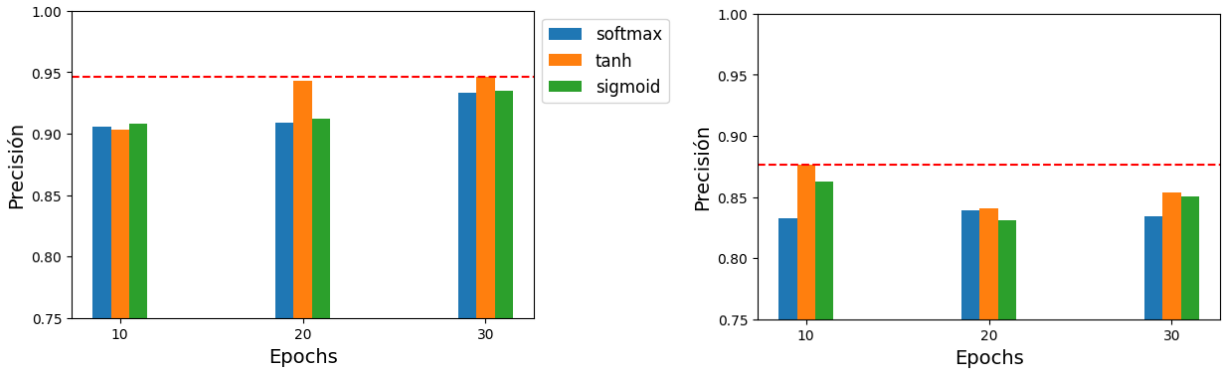
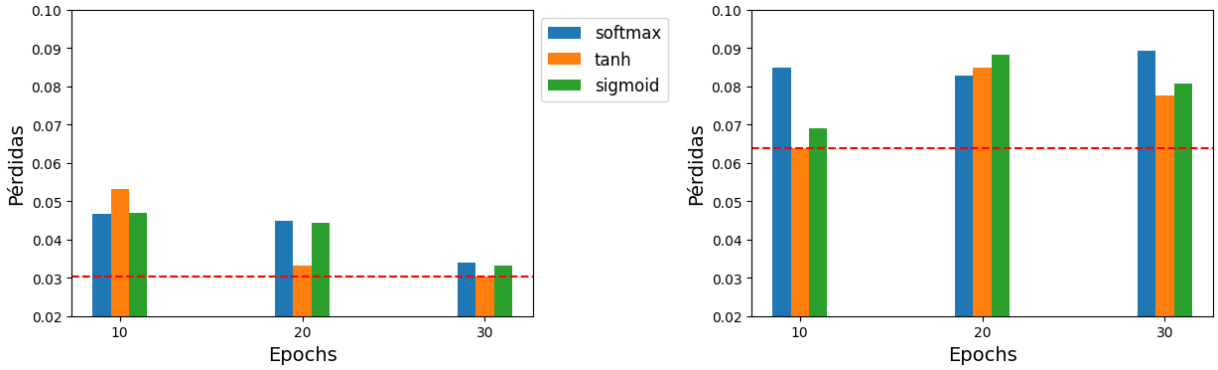


Figura 12

Pérdidas promedio para las LSTM con 16 neuronas entrenamiento (izq.) y evaluación (der.)



Los resultados en la evaluación para las LSTM con 16 neuronas se pueden ver de manera más precisa en la Tabla 4 junto con el tiempo de entrenamiento de estos 9 modelos juntos, mientras que las tablas con los resultados para los demás modelos se incluyen en el Apéndice E.

Tabla 4

Resultados de las LSTM con 16 neuronas

Epochs	**Función de activación**	**Precisión promedio de la evaluación**	**Pérdidas promedio (MSE) en la evaluación**
10	Softmax	83.28%	0.085
10	Tanh	87.62%	0.063
10	Sigmoid	86.23%	0.069
20	Softmax	83.91%	0.083
20	Tanh	84.09%	0.085
20	Sigmoid	83.12%	0.088
30	Softmax	83.38%	0.089
30	Tanh	85.34%	0.077
30	Sigmoid	85.08%	0.081
Tiempo de entrenamiento		29.5 min	

En el caso anterior, según la precisión, la función de activación Tanh mostró el mejor desempeño para las tres cantidades de *epochs* posibles. Además, la misma función presentó el

menor MSE para 10 y 30 *epochs*, pero no para 20. El mejor resultado se obtuvo para la combinación Tanh-10 *epochs*, con un valor de precisión del 87.62% y de error de 0.063, que fueron el mayor valor de precisión y el menor de error obtenidos de todos los modelos probados en este trabajo. Al observar las tablas (Apéndice E) con los resultados para números de neuronas superiores, se destaca que el tiempo de computación para el entrenamiento fue similar para los modelos con 16 y 32 neuronas (30 minutos aproximadamente), pero se duplicó progresivamente con cada aumento de neuronas a partir 64 sin representar una mejoría en los resultados.

Por otro lado, el análisis de las anteriores figuras y tabla junto con las incluidas en el Apéndice E permite observar que no hubo una función de activación que tuviera el mejor desempeño tanto en precisión como en pérdidas para todas las combinaciones de *epochs* y neuronas. En algunas ocasiones una función de activación tuvo el mejor resultado para una cantidad constante de neuronas, mientras que, en otras la función con el mejor desempeño varió al aumentar los *epochs* con el mismo número de neuronas.

En la Figura 13 se incluye la gráfica con los valores de precisión promedio, y en la Figura 14 con las pérdidas promedio, para los modelos con 10 *epochs*. Los mejores resultados, en términos de precisión más alta y MSE más bajo, se alcanzaron con 16 neuronas para las funciones Tanh y Sigmoid, mientras que para la función Softmax se lograron con 64 neuronas. Después de alcanzar el mejor valor posible para cada función, el aumento de neuronas solo representó un descenso en la precisión y un aumento en las pérdidas. Este análisis es respaldado por el trabajo de Mirzaei *et al* (2022), quienes obtuvieron un resultado óptimo de precisión con una LSTM de una sola capa y 16 neuronas, además, mostraron que un aumento en el número de neuronas conllevaba a una disminución en la precisión.

Figura 13

Precisión promedio en la evaluación de los modelos con 10 epochs

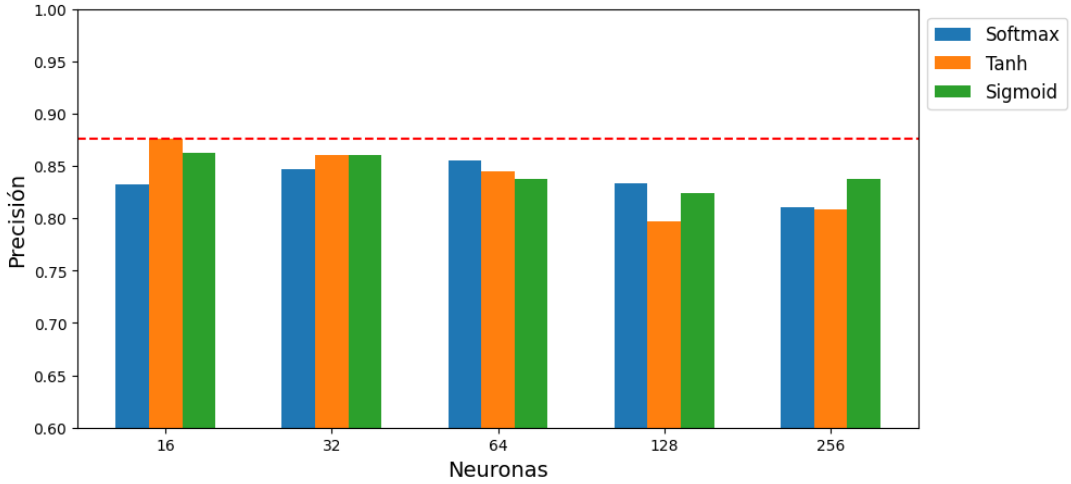
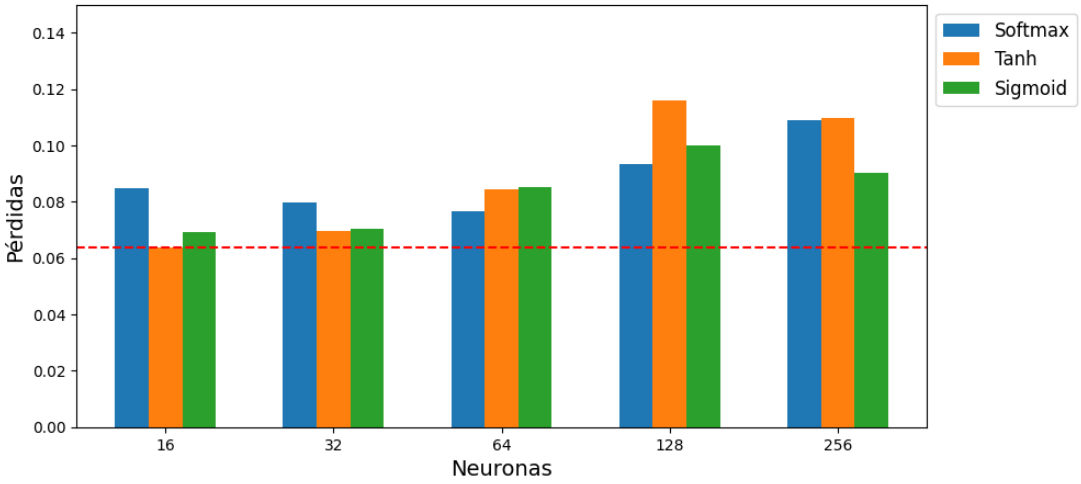


Figura 14

Pérdidas promedio en la evaluación de los modelos con 10 epochs



En el Apéndice F se incluyen las gráficas de precisión y pérdidas promedio en la evaluación de los modelos con números de *epochs* superiores a 10. Se observa nuevamente que los resultados empeoraron, la precisión disminuyó y el error aumentó a medida que se incrementó este parámetro. Además, se confirma también que el aumento del número de neuronas no representó una mejoría para los resultados.

En la Figura 15 se presenta la precisión para cada estado de operación (sin falla, falla 6 y falla 18) para la evaluación de la RNR recurrente tipo LSTM con 16 neuronas, 10 *epochs* y función de activación Tanh. Esta fue la arquitectura más eficiente y está representada en la Figura 16. La gráfica de la Figura 15 permite ver que la precisión fue bastante alta (superior al 95%) para la operación normal y la falla 18 pero media (entre 60 y 70%) para la falla 6.

Figura 15

Precisión en la evaluación de cada estado de operación para la arquitectura más eficiente

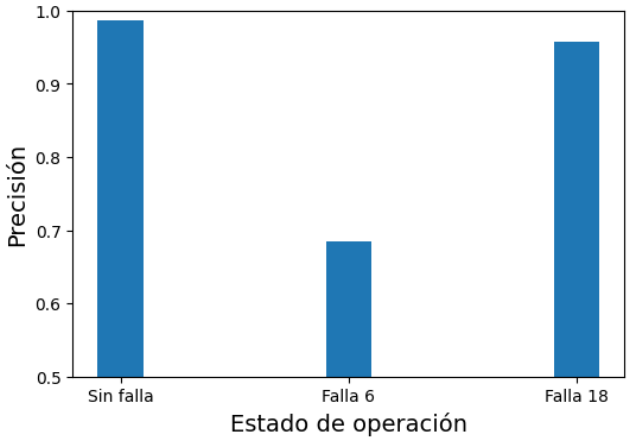
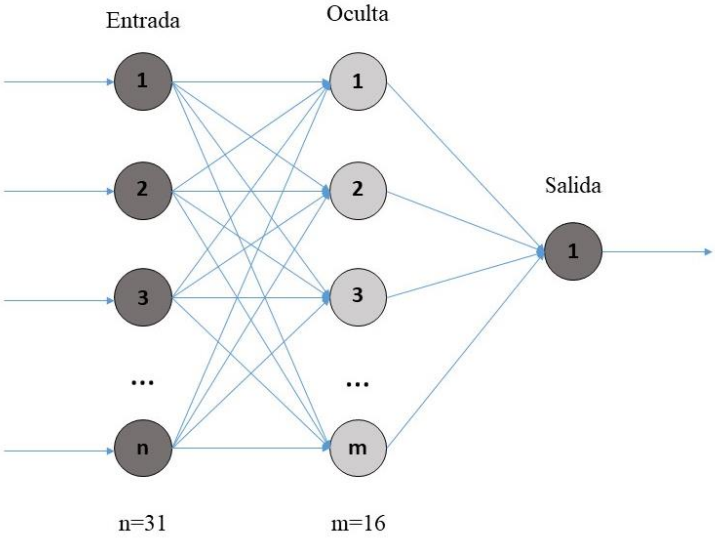


Figura 16

Esquema de la RNR tipo LSTM más eficiente

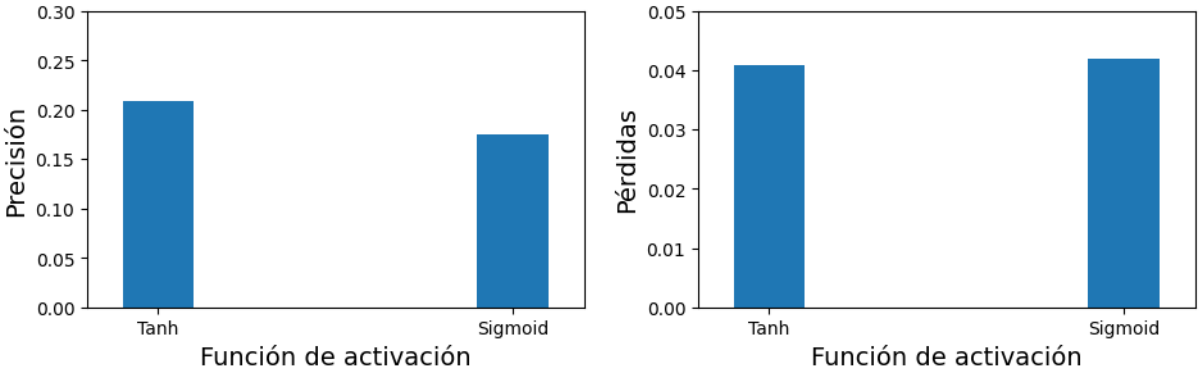


5.4 Detección y diagnóstico de todas las fallas

En la última etapa de experimentación, se pusieron a prueba los dos modelos que mostraron mejores resultados en la detección de las dos fallas críticas, la red neuronal LSTM de 16 neuronas y 10 *epochs* con las funciones de activación Tanh y Sigmoid, para la detección y el diagnóstico de las 20 fallas de las cuales se disponía información. Los demás parámetros, las características computacionales, la cantidad de datos por cada falla se mantuvieron igual que en los anteriores experimentos, asimismo el pretratamiento se realizó igual manteniendo la misma cantidad de componentes principales. Los resultados obtenidos para el modelo con la función Tanh fueron: una precisión promedio de 20.88% y un MSE de 0.0408; mientras que, para el modelo con la función Sigmoid fueron: una precisión promedio de 17.48% y un MSE de 0.0419. Estos resultados se pueden observar gráficamente en la Figura 17. Este resultado contrasta con el obtenido por Mirzaei *et al.* (2022) en su trabajo, quienes obtuvieron una precisión promedio de 95% en la detección de las 20 fallas con una red LSTM de 16 neuronas, sin embargo, algunas características como el método de pretratamiento, la cantidad de datos introducidos a la red, o algunos parámetros como los *epochs* variaron entre los dos experimentos.

Figura 17

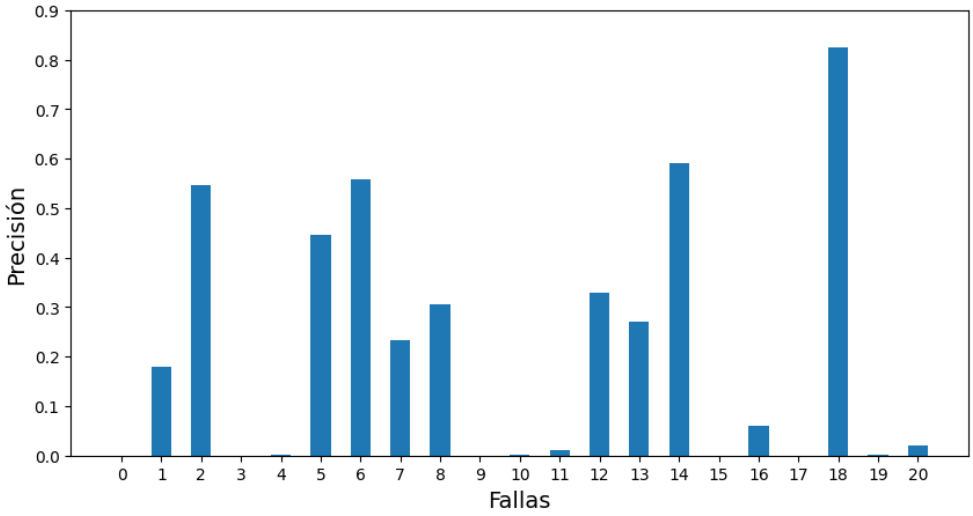
Resultados de las LSTM más eficientes en la detección de todas las fallas. Precisión (izq.) y pérdidas (der.)



Por último, la Figura 18 muestra la precisión en la detección de cada falla en la evaluación de la RNR con el mejor desempeño cuando fue evaluada con todas las fallas. El único resultado bueno fue el obtenido para la falla 18 (superior al 80%). La falla 6 obtuvo un resultado entre el 50 y 60%, mientras que, la operación sin falla (falla 0) obtuvo una precisión muy cercana al 0%. Las fallas 3, 9 y 15, conocidas como fallas incipientes, obtuvieron un resultado de 0. Este tipo de fallas suele ser eliminada de los estudios de detección y diagnóstico con redes neuronales debido a su similitud con la operación normal, lo que las hace de difícil detección y diagnóstico (González, 2020). Lo anterior explica por qué se obtuvo una precisión de 0 para esos cuatro estados de operación y es una de las razones por las cuales al entrenar y evaluar la red con todas las fallas los resultados fueron deficientes. Dicho de otro modo, el modelo más eficiente para la detección de las dos fallas críticas no hace buenas predicciones cuando se intenta usar para detectar y diagnosticar todas las fallas.

Figura 18

Precisión en la evaluación de todas las fallas con la arquitectura más eficiente



6. Conclusiones

Según los resultados del pretratamiento, las 52 variables del proceso Tennessee Eastman pueden ser reducidas a los primeros 31 componentes principales para la falla 6 y 18 manteniendo una reproducción de la varianza de más del 90%.

La red neuronal recurrente tipo LSTM puede ser codificada mediante las rutinas de Python en el ambiente *Google Colab*. Considerando los parámetros *epochs* (10, 20, 30), neuronas (16, 32, 64, 128, 256) y las funciones de activación (Tanh, Sigmoid, Softmax) entre los 45 modelos probados se obtuvo una precisión promedio en la evaluación de la red mínima de 76.82% y una máxima de 87.62%, la mayoría de los modelos tuvieron un resultado superior al 80% en esta métrica. Las pérdidas promedio de la evaluación de la red estuvieron en un rango de 0.063 a 0.138 y más de la mitad de los experimentos tuvieron un error menor al 0.1.

También, se pudo comprobar que el aumento de *epochs* en el rango trabajado (10-30) no representa una mejoría en los resultados; la función de activación de salida de la red con mejor

desempeño varía dependiendo de los otros parámetros. Además, se pudo corroborar que aumentar el número de neuronas perjudica los resultados y 16 es el número óptimo de estas para detectar y clasificar las fallas 6 y 18 del Tennessee Eastman con este tipo de red neuronal.

La estructura más eficiente entre las RNR evaluadas, es decir la que tiene mayor precisión promedio en la evaluación con un valor de 87.62% y menores pérdidas promedio en la evaluación con un MSE de 0.063 para la detección de las fallas 6 y 18, fue la arquitectura que combinó los parámetros: 16 neuronas-Tanh-10 *epochs*.

Como conclusión general, las redes neuronales recurrentes tipo LSTM tienen una precisión promedio alrededor del 80% en la detección y diagnóstico de las fallas 6 y 18 del proceso Tennessee Eastman, por lo cual muestran su potencial aplicación en el ámbito industrial.

7. Recomendaciones

Se recomienda para futuras investigaciones, intentar como pretratamiento otros métodos de reducción de dimensionalidad como ICA y t-SNE que sean más apropiados para datos altamente no lineales. También, se aconseja determinar si las métricas de desempeño de la RNR tipo LSTM mejoran al utilizar una cantidad diferente de simulaciones de los estados de operación del TEP (más o menos de 50), al variar otros parámetros como, la función de activación de las neuronas de la capa interna, el *batch size*, o probar con un rango mayor de *epochs* (más de 30). Como trabajo futuro, también se podría usar la red *General recurrent unit* (GRU), otro tipo de red neuronal recurrente, y determinar si es más eficiente que la LSTM, o experimentar con alguna versión mejorada de esta última, tal como, la bidireccional.

Referencias bibliográficas

Amibp. (2023, 7 junio). *Ajuste de hiperparámetros de un modelo (v2) - Azure Machine Learning*. Microsoft Learn. https://learn.microsoft.com/es-es/azure/machine-learning/how-to-tune-hyperparameters?view=azureml-api-2

apsl.net. (s. f.). *Análisis de series temporales usando redes neuronales recurrentes*. https://apsl.tech/es/blog/analisis-de-series-temporales-usando-redes-neuronales-recurrentes/

Arunthavanathan, R., Khan, F., Ahmed, S., Imtiaz, S., & Rusli, R. (2020). Fault detection and diagnosis in process system using artificial intelligence-based cognitive technique. *Computers & Chemical Engineering*, *134*, 106697. https://doi.org/10.1016/j.compchemeng.2019.106697

Ayodeji, A., Liu, Y., & Xia, H. (2018). Knowledge base operator support system for nuclear power plant fault diagnosis. *Progress In Nuclear Energy*, *105*, 42-50. https://doi.org/10.1016/j.pnucene.2017.12.013

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Cárdenas, J., & Reyes, S. (2021). *Detección de fallas operacionales con redes neuronales artificiales: aplicación del proceso Tennessee Eastman* [Tesis de pregrado]. Universidad Industrial de Santander.

Casas, J. M. (2020, 23 junio). *Tuneando los hiperparámetros de una red neuronal LSTM para obtener un aprendizaje más eficiente*. https://es.linkedin.com/pulse/tuneando-los-hiperpar%C3%A1metros-de-una-red-neuronal-lstm-casas-

gonzalez#:~:text=Batch%20Size%3A%20este%20es%20el,que%20entrene%20de%20cada%20vez.

Chiang, L. H., Russell, E. L., & Braatz, R. D. (2001a). Fault Detection and Diagnosis in Industrial Systems. En *Advanced textbooks in control and signal processing*. https://doi.org/10.1007/978-1-4471-0347-9

Chiang, L. H., Russell, E. L., & Braatz, R. D. (2001b). Fault Detection and Diagnosis in Industrial Systems. *Measurement Science And Technology*, *12*(10), 1745. https://doi.org/10.1088/0957-0233/12/10/706

Ciberseg. (2021, 15 diciembre). *¿Qué es Epoch en Machine Learning?* Ciberseguridad. https://ciberseguridad.com/guias/nuevas-tecnologias/machine-learning/epoch/#:~:text=Epoch%20en%20la%20red%20neuronal,-Una%20red%20neuronal&text=De%20manera%20similar%20al%20uso,conjunto%20de%20datos%20de%20entrenamiento

Cruz, I., Salazar, S., Rodríguez, A., Grau, R., & García, M. (2007). Redes neuronales recurrentes para el análisis de secuencias. *Revista Cubana de Ciencias Informáticas*, *1*(4).

Curiel, J. (2022). *Mejora de la calidad de un proceso mediante la detección de anomalías basada en inteligencia artificial computacional* [Grado en ingeniería electrónica industrial y automático]. Universidad de Valladolid.

Downs, J. J., & Vogel, E. F. (1993). A plant-wide industrial process control problem. *Computers & Chemical Engineering*, *17*(3), 245-255. https://doi.org/10.1016/0098-1354(93)80018-i

González, M. (2020). *Estudio de técnicas de clasificación para detección y diagnóstico de fallos* [Grado en organización industrial]. Universidad de Valladolid.

Hajihosseini, P., Anzehaee, M. M., & Behnam, B. (2018). Fault detection and isolation in the challenging Tennessee Eastman process by using image processing techniques. *ISA Transactions*, *79*, 137-146. https://doi.org/10.1016/j.isatra.2018.05.002

Kang, J. (2020). Visualization analysis for fault diagnosis in chemical processes using recurrent neural networks. *Journal Of The Taiwan Institute Of Chemical Engineers*, *112*, 137-151. https://doi.org/10.1016/j.jtice.2020.06.016

Kostadinov, S. (2018). *Recurrent neural networks with Python quick start guide: sequential learning and language modeling with TensorFlow* (1st edition) [Packt].

Lyman, P. R., & Georgakis, C. (1995). Plant-wide control of the Tennessee Eastman problem. *Computers & Chemical Engineering*, *19*(3), 321-331. https://doi.org/10.1016/0098-1354(94)00057-u

Medrano, J. (2019). *Estudio de técnicas de clasificación para la detección y diagnóstico de fallos* [Grado en Ingeniería Química]. Universidad de Valladolid.

Melo, A., Câmara, M. M., & Pinto, J. C. (2024). Data-Driven Process Monitoring and Fault Diagnosis: A Comprehensive Survey. *Processes*, *12*(2), 251. https://doi.org/10.3390/pr12020251

Mirzaei, S., Kang, J., & Chu, K. (2022). A comparative study on long short-term memory and gated recurrent unit neural networks in fault diagnosis for chemical processes using visualization. *Journal Of The Taiwan Institute Of Chemical Engineers*, *130*, 104028. https://doi.org/10.1016/j.jtice.2021.08.016

Pequeño, Á. (2020). *Mejora del control de calidad de un proceso mediante técnicas de aprendizaje automático* [Grado en Ingeniería Química]. Universidad de Valladolid.

Redes neuronales de memoria de corto-largo plazo - MATLAB & Simulink - MathWorks América Latina. (s. f.). https://la.mathworks.com/help/deeplearning/ug/long-short-term-memory-networks.html

Rieth, C. A., Amsel, B. D., Tran, R., & Cook, M. B. (2017). *Additional Tennessee Eastman Process Simulation Data for Anomaly Detection Evaluation* [Conjunto de datos]. https://doi.org/10.7910/dvn/6c3jr1

Rué, A. B., Roma, J. C., & Bagén, T. L. (2019). *Deep learning: principios y fundamentos*.

Salas, R. (2004). Redes Neuronales Artificiales. *Universidad de Valparaiso. Departamento de Computación*, 1-7.

Salem, F. M. (2022). *Recurrent Neural Networks: From Simple to Gated Architectures.* Suiza: Springer International Publishing.

Sheng, X. (2020). *Machine learning techniques for fault detection in chemical processes. The Tennessee Eastman Process case study* [Tesis de maestría]. Politécnico Di Milano.

Sheta, A., Braik, M. S., & Al-Hiary, H. (2009). Identification and model predictive controller design of the Tennessee Eastman Chemical Process using ANN. *ResearchGate*. https://www.researchgate.net/publication/220834404_Identification_and_Model_Predictive_Controller_Design_of_the_Tennessee_Eastman_Chemical_Process_Using_ANN

Tech, T. (s. f.). *¿Qué es la Función de Activación?* ¿Qué Es la Función de Activación? https://aiofthings.telefonicatech.com/recursos/datapedia/funcion-

activacion#:~:text=A%20la%20salida%20de%20la,conoce%20como%20funci%C3%B3n%20de%20activaci%C3%B3n

Yin, S., Ding, S. X., Haghani, A., Hao, H., & Zhang, P. (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), 1567-1581. https://doi.org/10.1016/j.jprocont.2012.06.009

Apéndices

Apéndice A

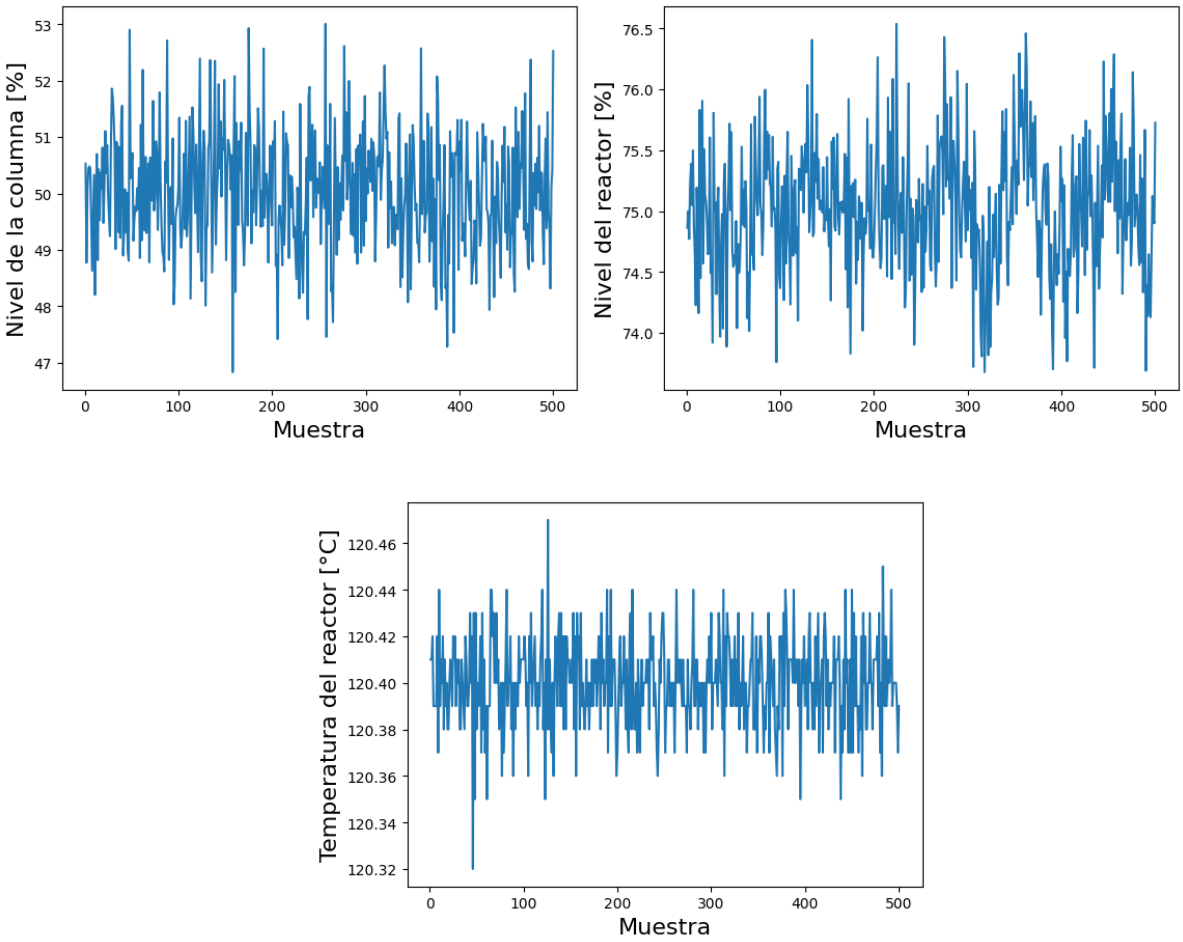
Descripción de las fallas del TEP

Fallo	Variables de fallo	Tipo de fallo
IDV (1)	Relación de flujo de alimentaciones A/C, composición de B constante	Escalón
IDV (2)	Composición de B con relación A/C constante	Escalón
IDV (3)	Temperatura de alimentación D	Escalón
IDV (4)	Temperatura de entrada del agua refrigerante del reactor	Escalón
IDV (5)	Temperatura de entrada del agua refrigerante del condensador	Escalón
IDV (6)	Pérdida de alimentación de A	Escalón
IDV (7)	Pérdida de presión en la corriente C	Variación aleatoria
IDV (8)	Composición de las alimentaciones A, B y C	Variación aleatoria
IDV (9)	Temperatura de alimentación D	Variación aleatoria
IDV (10)	Temperatura de alimentación C	Variación aleatoria
IDV (11)	Temperatura de entrada del agua refrigerante del reactor	Variación aleatoria
IDV (12)	Temperatura de entrada del agua refrigerante del condensador	Variación aleatoria
IDV (13)	Cinética de las reacciones	Variación lenta
IDV (14)	Válvula de agua refrigerante del reactor	Bloqueo
IDV (15)	Válvula de agua refrigerante del condensador	Bloqueo
IDV (16)	Desconocido	No especificado
IDV (17)	Desconocido	No especificado
IDV (18)	Desconocido	No especificado
IDV (19)	Desconocido	No especificado
IDV (20)	Desconocido	No especificado
IDV (21)	Desconocido	Constante

Apéndice B *Comportamiento de las variables importantes sin falla*

Figura 19

Comportamiento de las variables importantes sin falla



Apéndice C *Varianza de los componentes principales.*

Figura 20

Varianza representada por los componentes principales para los datos de evaluación sin falla

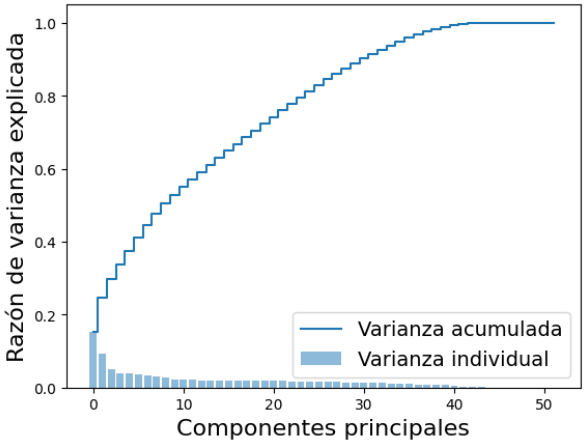
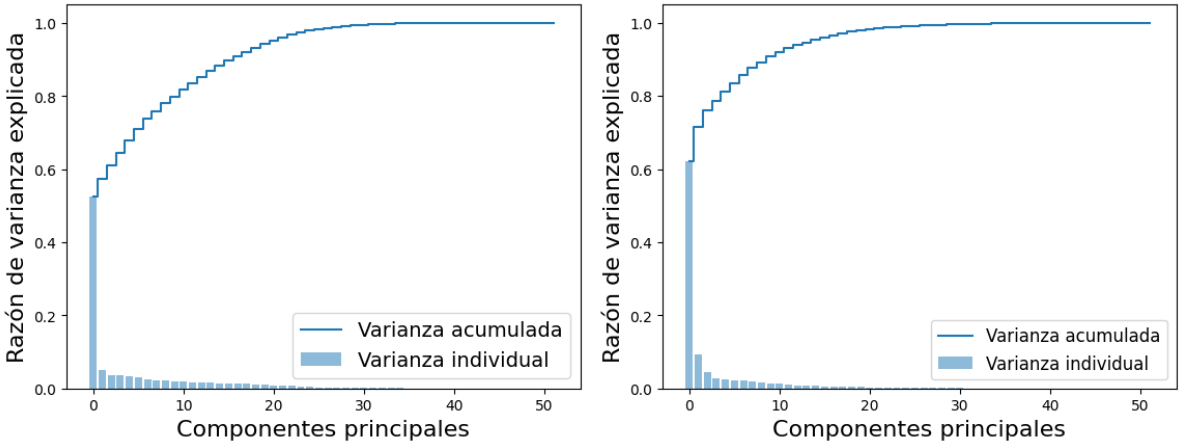


Figura 21

Varianza representada por los componentes principales para los datos de evaluación de la falla 6 (izq.) y 18 (der.)



Apéndice D *Comportamiento de las variables importantes con falla*

Figura 22

Comportamiento de las variables importantes con la falla 6

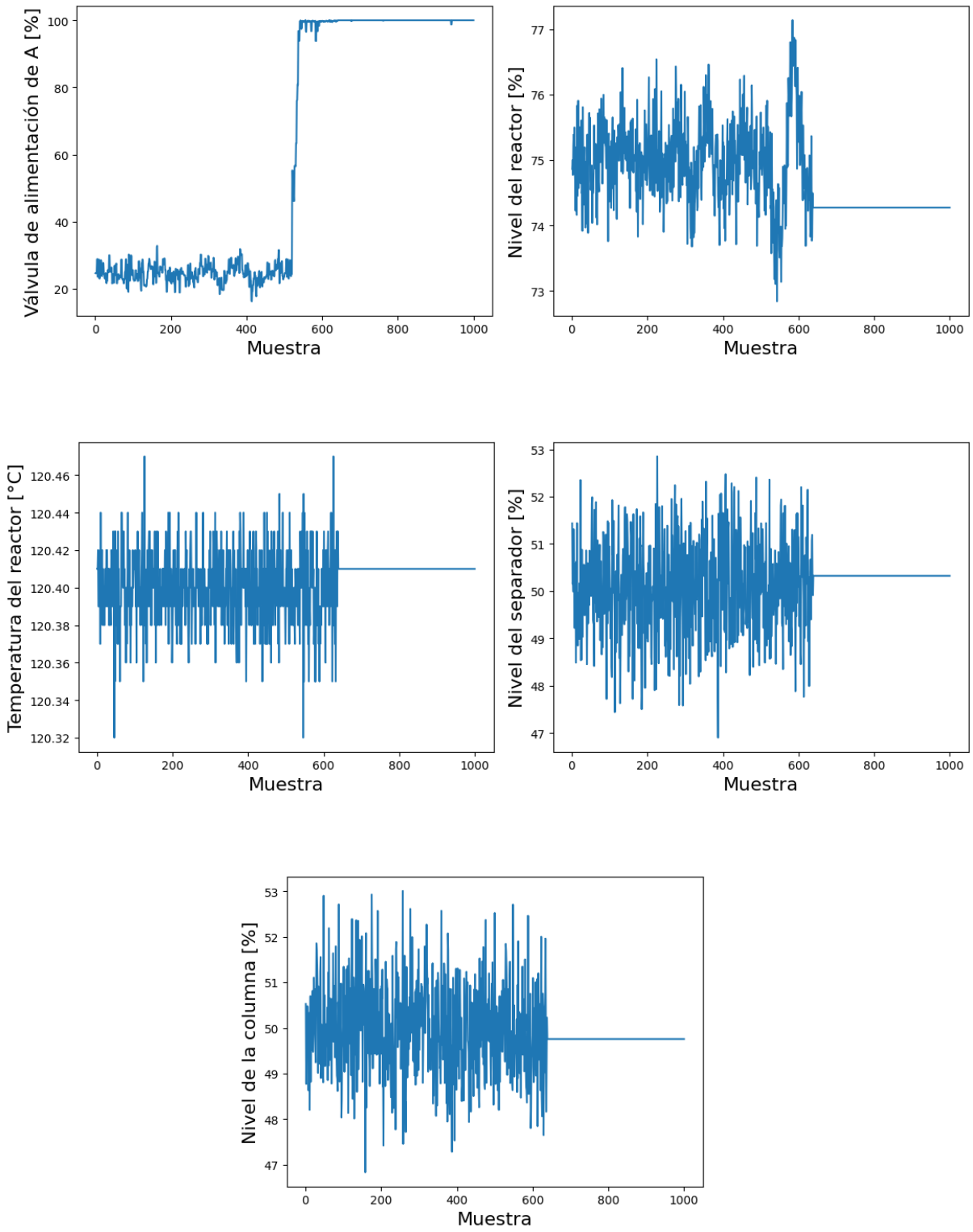
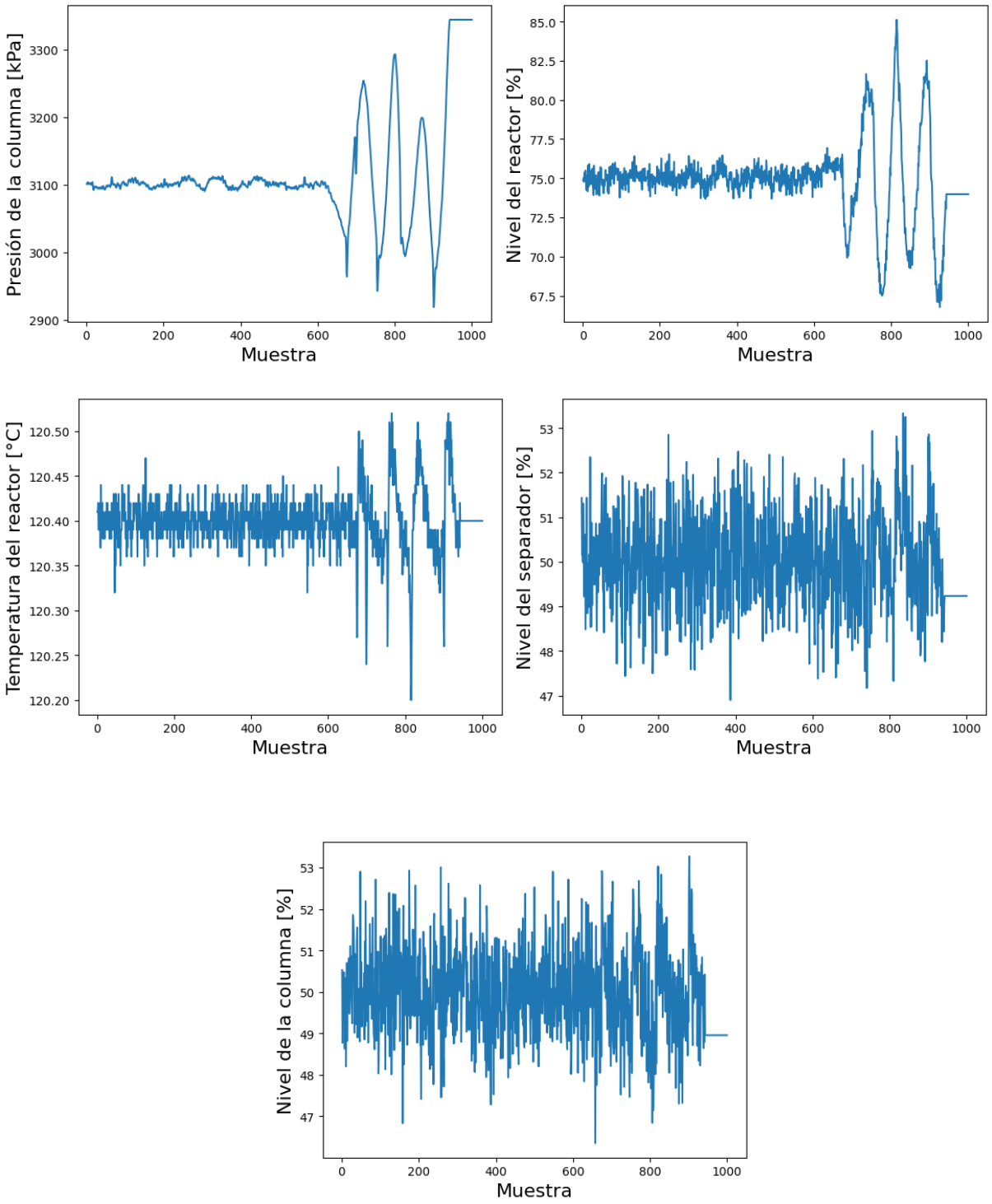


Figura 23

Comportamiento de las variables importantes con la falla 18



Apéndice E *Resultados para las RNR-LSTM*

Figura 24

Precisión promedio para las LSTM con 32 neuronas entrenamiento (izq.) y evaluación (der.)

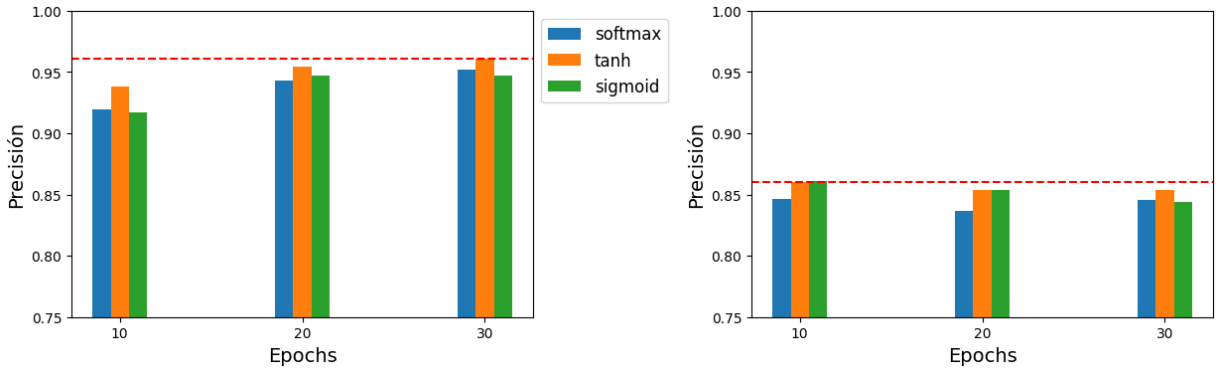


Figura 25

Pérdidas promedio para las LSTM con 32 neuronas entrenamiento (izq.) y evaluación (der.)

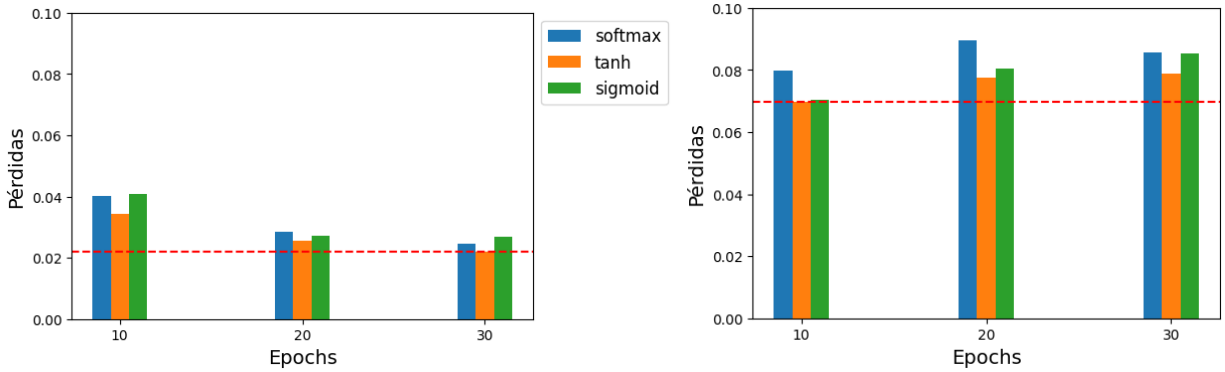


Figura 26

Precisión promedio para las LSTM con 64 neuronas entrenamiento (izq.) y evaluación (der.)

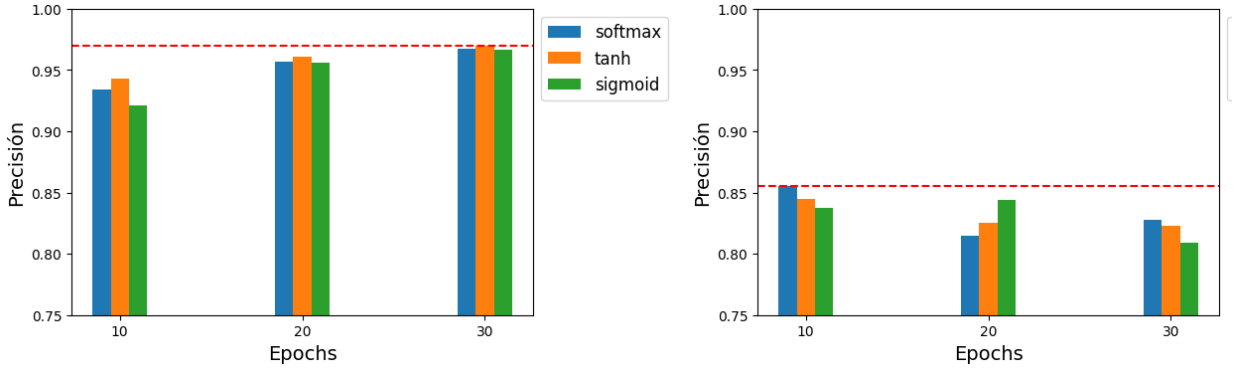


Figura 27

Pérdidas promedio para las LSTM con 64 neuronas entrenamiento (izq.) y evaluación (der.)

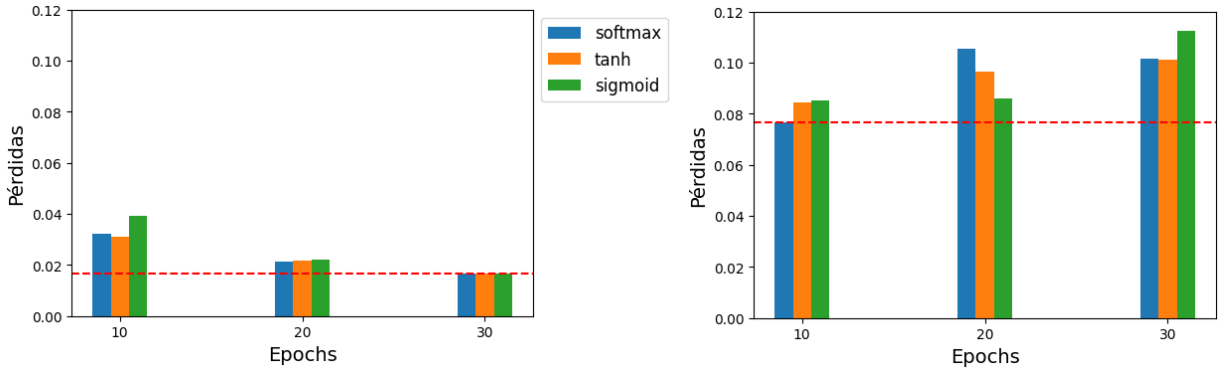


Figura 28

Precisión promedio para las LSTM con 128 neuronas entrenamiento (izq.) y evaluación (der.)

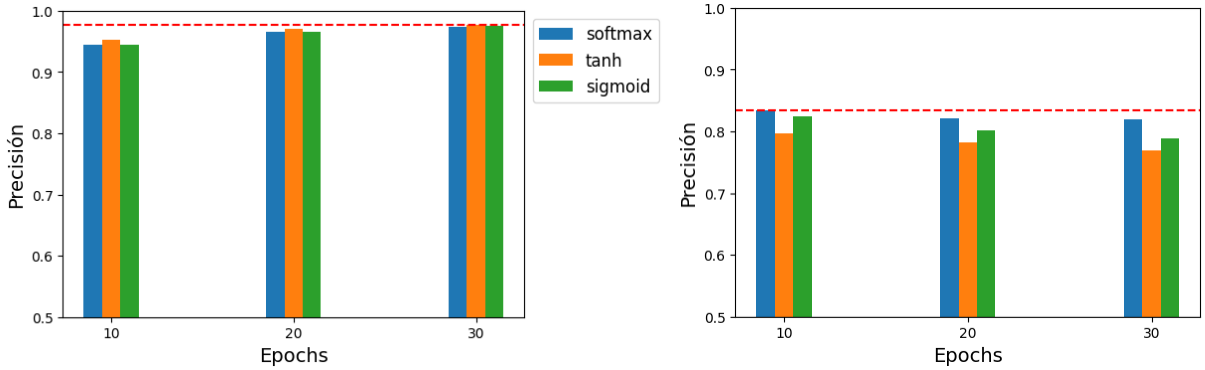


Figura 29

Pérdidas promedio para las LSTM con 128 neuronas entrenamiento (izq.) y evaluación (der.)

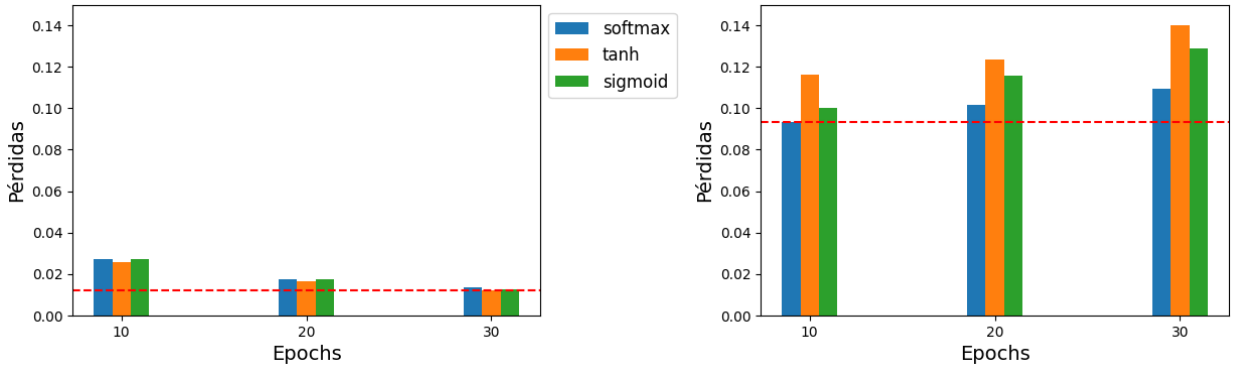


Figura 30

Precisión promedio para las LSTM con 256 neuronas entrenamiento (izq.) y evaluación (der.)

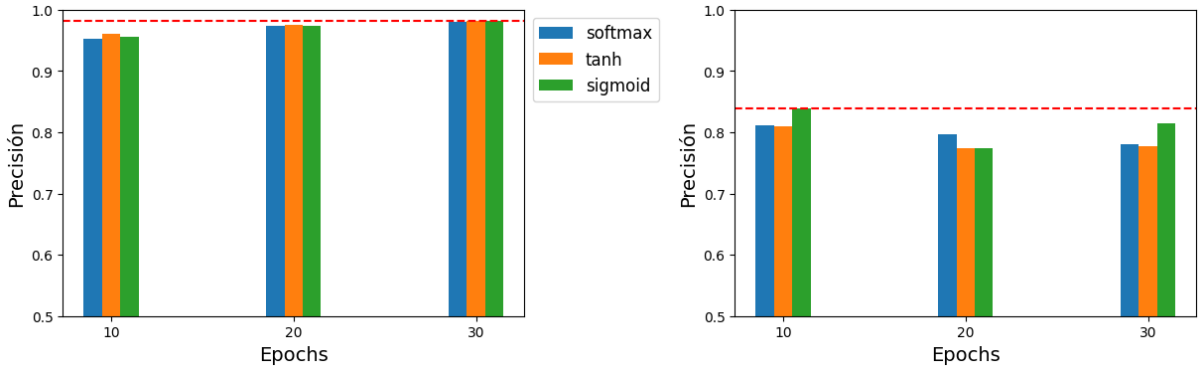


Figura 31

Pérdidas promedio para las LSTM con 256 neuronas entrenamiento (izq.) y evaluación (der.)

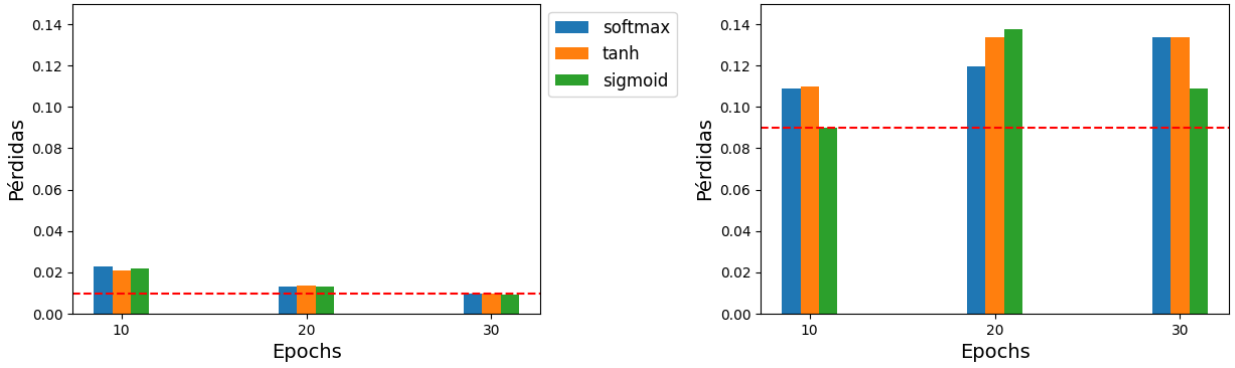


Tabla 5

Resultados de las LSTM con 32 neuronas

Epochs	**Función de activación**	**Precisión promedio de la evaluación**	**Pérdidas promedio (MSE) en la evaluación**
10	Softmax	84.66%	0.079
10	Tanh	86.00%	0.069
10	Sigmoid	86.09%	0.070
20	Softmax	83.69%	0.089
20	Tanh	85.40%	0.077
20	Sigmoid	85.37%	0.080
30	Softmax	84.54%	0.085
30	Tanh	85.35%	0.079
30	Sigmoid	84.41%	0.085
Tiempo de entrenamiento		31.9 min	

Tabla 6

Resultados de las LSTM con 64 neuronas

Epochs	**Función de activación**	**Precisión promedio de la evaluación**	**Pérdidas promedio (MSE) en la evaluación**
10	Softmax	85.51%	0.076
10	Tanh	84.46%	0.084
10	Sigmoid	83.75%	0.085
20	Softmax	81.47%	0.105
20	Tanh	82.55%	0.097
20	Sigmoid	84.41%	0.085
30	Softmax	82.76%	0.101
30	Tanh	82.27%	0.101
30	Sigmoid	80.89%	0.112
Tiempo de entrenamiento		58.7 min	

Tabla 7

Resultados de las LSTM con 128 neuronas

Epochs	**Función de activación**	**Precisión promedio de la evaluación**	**Pérdidas promedio (MSE) en la evaluación**
10	Softmax	83.33%	0.093
10	Tanh	79.66%	0.116
10	Sigmoid	82.39%	0.100
20	Softmax	82.13%	0.101
20	Tanh	78.14%	0.123
20	Sigmoid	80.09%	0.116
30	Softmax	81.89%	0.109
30	Tanh	76.82%	0.140
30	Sigmoid	78.84%	0.128
Tiempo de entrenamiento		88.9 min	

Tabla 8

Resultados de las LSTM con 256 neuronas

Epochs	**Función de activación**	**Precisión promedio de la evaluación**	**Pérdidas promedio (MSE) en la evaluación**
10	Softmax	81.06%	0.109
10	Tanh	80,89%	0.110
10	Sigmoid	83.80%	0.090
20	Softmax	79.64%	0.120
20	Tanh	77.35%	0.134
20	Sigmoid	77.36%	0.138
30	Softmax	78.01%	0.134
30	Tanh	77.66%	0.134
30	Sigmoid	81.50%	0.109
Tiempo de entrenamiento		186.7 min	

Apéndice F *Comparación de resultados con diferentes epochs*.

Figura 32

Precisión en la evaluación de los modelos con 20 epochs

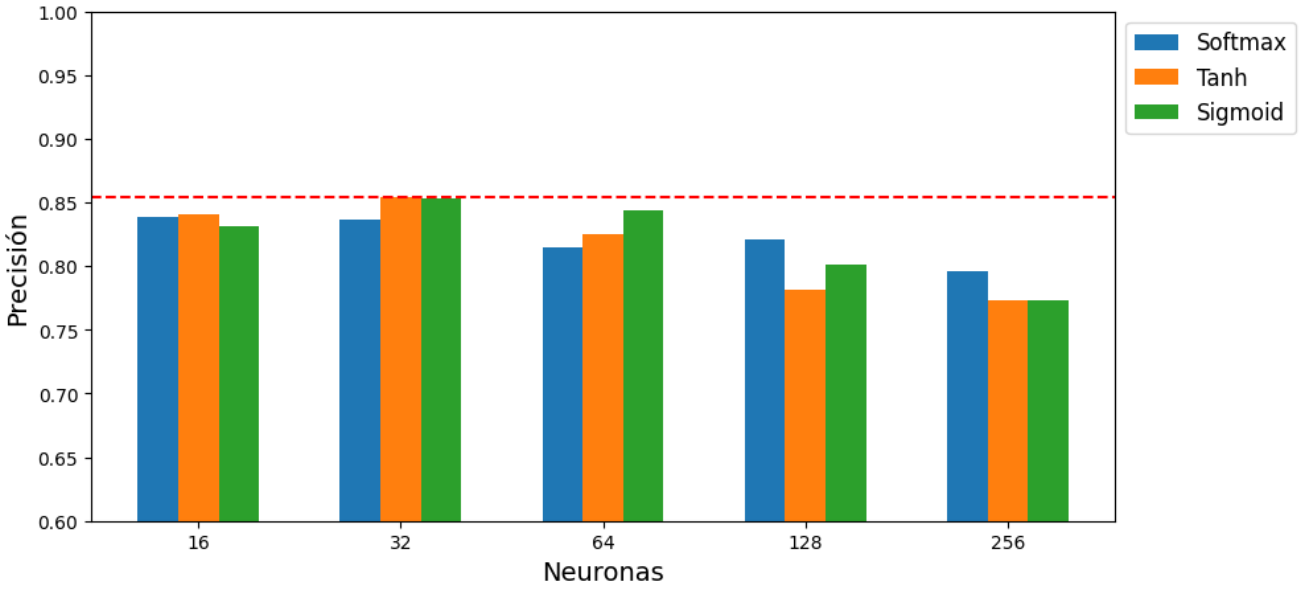


Figura 33

Precisión en la evaluación de los modelos con 30 epochs

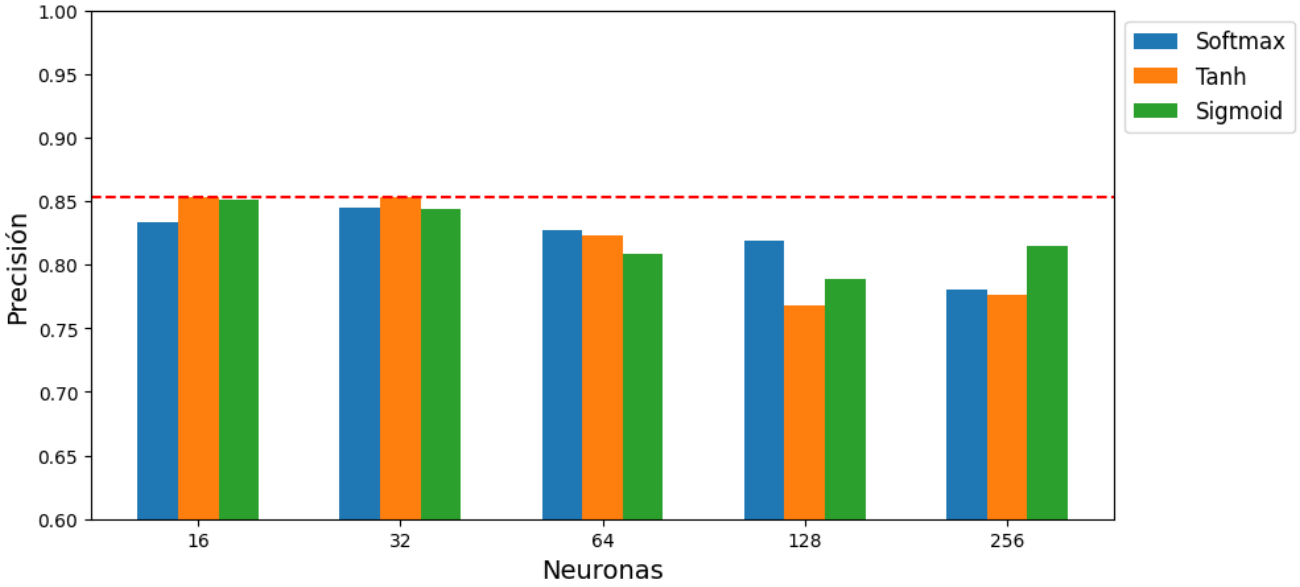


Figura 34

Pérdidas en la evaluación de los modelos con 20 epochs

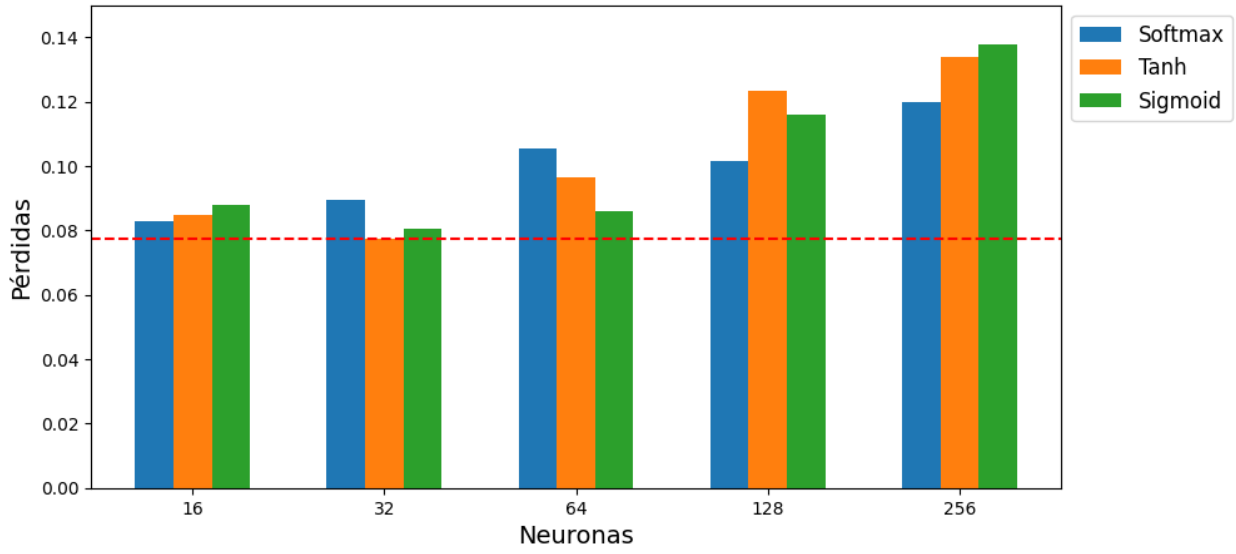
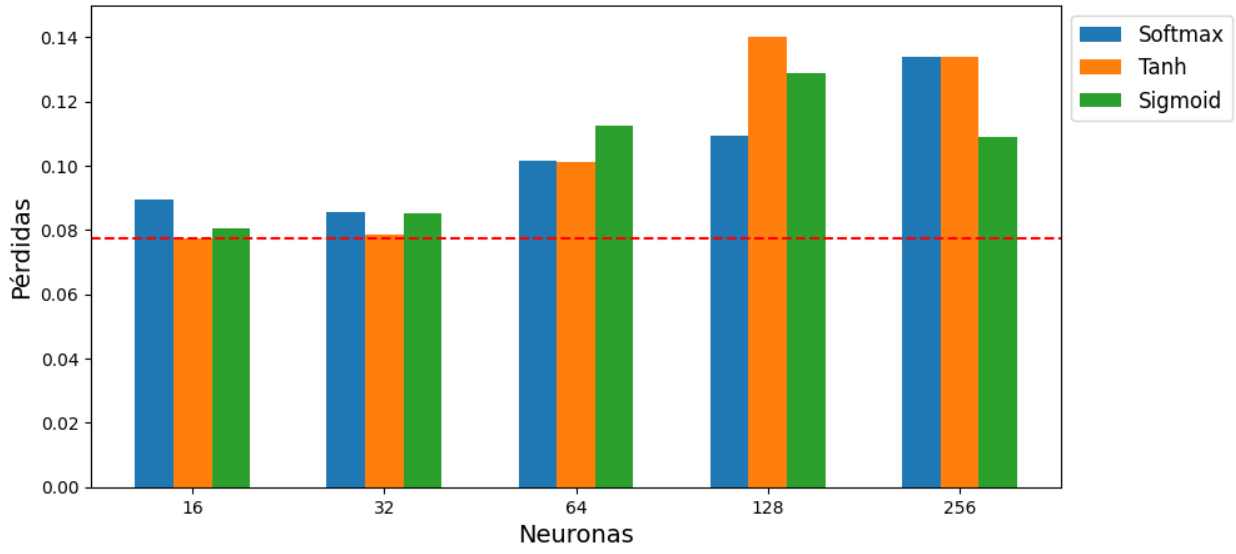


Figura 35

Pérdidas en la evaluación de los modelos con 30 epochs



Apéndice G *Código en Python para el pretratamiento de los datos*

```
# Instaladores
!pip install pyreadr
!pip install --upgrade scikit-learn

#Importación de librerías
import pyreadr
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
from google.colab import drive
drive.mount('/content/drive')#Acceso a drive

#Apertura de los ficheros
df_ff = pyreadr.read_r('/content/drive/MyDrive/Tesis/TEP_FaultFree_Training.RData')['fault_free_training']
df_fy = pyreadr.read_r('/content/drive/MyDrive/Tesis/TEP_Faulty_Training.RData')['faulty_training']
dff_test= pyreadr.read_r('/content/drive/MyDrive/Tesis/TEP_FaultFree_Testing.RData')['fault_free_testing']
dfy_test=pyreadr.read_r('/content/drive/MyDrive/Tesis/TEP_Faulty_Testing.RData')['faulty_testing']

#Creación de una nueva simulación con más datos

df6_e = pd.concat([df_ff[df_ff["simulationRun"] == 1], df_fy[(df_fy["faultNumber"] == 6)
 & (df_fy["simulationRun"] == 1)]])

#Aplicación del estadístico Hotteling
hotteling_transform=pca6_e.components_.T
X_hotteling=df_pca6_e
T2 = np.sum(X_hotteling**2, axis=1)
mu = np.mean(T2)
std = np.std(T2)
T2_threshold = mu + std
```

```
#Normalizar los datos de entrenamiento de la falla 6

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
f6=df_fy[df_fy["faultNumber"]==6]
fault_6 =f6[f6["simulationRun"]<=50].iloc[:,3:]
df_scaled6 = sc.fit_transform(fault_6)

#Normalizar los datos de evaluación de la falla 6

sc = StandardScaler()
f6=dfy_test[dfy_test["faultNumber"]==6]
fault_6 =f6[f6["simulationRun"]<=50].iloc[:,3:]
df_scaled6_t = sc.fit_transform(fault_6)

#Matriz de covarianza

cov_mat = np.cov(df_scaled6_t.T)
eigen_vals, eigen_vecs = np.linalg.eig(cov_mat)


#Transformación de los datos de entrenamiento y evaluación mediante el PCA

from sklearn.decomposition import PCA

pca6 = PCA(n_components=31)
df_pca6 = pca6.fit_transform(df_scaled6)
pca6_t=PCA(n_components=31)
df_pca6_t=pca6_t.fit_transform(df_scaled6_t)
df_pca6_t
```

Apéndice H *Código en Python para las redes neuronales recurrentes tipo LSTM.*

```python
#Creación de x_train y y_train
import numpy as np
import tensorflow as tf
x_train=np.concatenate((df_0trai,df_6trai,df_18trai),axis=0)
y_train = []
for i in range(df_0trai.shape[0]):
    y_train.append(0)
for i in range(df_6trai.shape[0]):
    y_train.append(1)
for i in range(df_18trai.shape[0]):
    y_train.append(2)
y_train = np.array(y_train)
y_train = tf.keras.utils.to_categorical(y_train, num_classes=3)

#Separación de los datos de validación a partir de los de evaluación
from sklearn.model_selection import train_test_split
valid_ff, test_ff = train_test_split(df_0test, test_size=0.8, random_state=23)
valid_f6, test_f6 = train_test_split(df_6test, test_size=0.8, random_state=23)
valid_f18, test_f18 = train_test_split(df_18test, test_size=0.8, random_state=23)

#Creación de x_valid y y_valid
x_valid=np.concatenate((valid_ff,valid_f6,valid_f18),axis=0)
y_valid = []
for i in range(valid_ff.shape[0]):
    y_valid.append(0)
for i in range(valid_f6.shape[0]):
    y_valid.append(1)
for i in range(valid_f18.shape[0]):
    y_valid.append(2)
y_valid = np.array(y_valid)
y_valid = tf.keras.utils.to_categorical(y_valid, num_classes=3)
```

```python
#Importación de librerías
import time
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.models import Sequential

#Creación de la red LSTM con 16 neuronas
start_time = time.time()
def train_and_evaluate_lstm(epochs, activation):
    model = Sequential()
    model.add(LSTM(units=16, input_shape=(31,1)))
    model.add (Dense(units=3, activation=activation))
    model.compile(loss='mse', optimizer='adam', metrics=['accuracy'])

#Entrenamiento de la red
    history = model.fit(x_train, y_train, epochs=epochs, batch_size=128, validation_data=(x_valid, y_valid))
    return history
epochs_list = [10]
activation_list = ['tanh']
results = []
for epochs in epochs_list:
  for activation in activation_list:
    history= train_and_evaluate_lstm(epochs, activation)
    results.append({'epochs': epochs, 'activation': activation, 'history': history})

#Cálculo del tiempo de entrenamiento
end_time = time.time()
training_time = end_time - start_time
print(f"Training time: {training_time:.2f} seconds")
```

```
#Obtención de precisión y pérdidas promedio para el entrenamiento y la validación

average_results = []
for result in results:
    history = result['history']
    loss = history.history['loss']
    accuracy = history.history['accuracy']
    val_loss = history.history['val_loss']
    val_accuracy = history.history['val_accuracy']

    train_loss = sum(loss) / len(loss)
    train_accuracy = sum(accuracy) / len(accuracy)
    val_loss = sum(val_loss) / len(val_loss)
    val_accuracy = sum(val_accuracy) / len(val_accuracy)

    average_results.append({'epochs': result['epochs'], activation': result['activation'],
        'train_loss': train_loss,'train_accuracy': train_accuracy,'val_loss': val_loss,
        'val_accuracy': val_accuracy})

#Creación de x_test y y_test

x_test=np.concatenate((test_ff,test_f6,test_f18),axis=0)
y_test= []

for i in range(test_ff.shape[0]):
    y_test.append(0)
for i in range(test_f6.shape[0]):
    y_test.append(1)
for i in range(test_f18.shape[0]):
    y_test.append(2)
```

```
y_test = np.array(y_test)
y_test = tf.keras.utils.to_categorical(y_test, num_classes=3)Test_results={}

#Evaluación de la LSTM
for result in results:
    model = result['history'].model
    test_loss, test_accuracy = model.evaluate(x_test, y_test)

#Creación de un diccionario para guardar los resultados
    model_results = {
        "Epochs": result['epochs'],
        "Activation": result['activation'],
        "Test Loss": test_loss,
        "Test Accuracy": test_accuracy,}

#Impresión de los resultados
    Test_results[model]=model_results
    print(f"Model: {result['epochs']} epochs, {result['activation']} activation")
    print(f"Test loss: {test_loss}")
    print(f"Test accuracy: {test_accuracy}")
    print()
```