RIELAC
Revista de Ingeniería Electrónica, Automática y Comunicaciones

Un modelo LSTM optimizado para el monitoreo de condiciones en procesos químicos

Adrián Rodríguez Ramos

RESUMEN / *ABSTRACT*

Los sistemas avanzados de monitorización de condiciones son esenciales para mejorar la eficiencia, la seguridad y la ciberseguridad en los procesos químicos, especialmente en la Industria 4.0. Este trabajo propone un modelo LSTM optimizado para la detección y localización simultáneas de fallos y ciberataques. El aprendizaje profundo facilita el procesamiento robusto de conjuntos de datos complejos y con ruido, mejorando el rendimiento de la clasificación incluso bajo condiciones de fallos y ataques superpuestos. La metodología incluye una fase fuera de línea para el etiquetado y la normalización de datos, así como la optimización del modelo LSTM mediante el algoritmo Evolución Diferencial. Gracias a esta optimización, se adquiere la robustez necesaria para gestionar los desafíos de los datos industriales, incluyendo el ruido y las perturbaciones externas. Durante el funcionamiento en línea, el sistema clasifica las observaciones en tiempo real en condiciones normales de funcionamiento, fallos o ciberataques, identificando condiciones desconocidas para el análisis experto y el reentrenamiento del sistema. Evaluado mediante el proceso de referencia de Tennessee Eastman (TE), el marco propuesto demuestra una precisión y robustez superiores a las de los métodos existentes, particularmente en escenarios complejos como los fallos 9 y 15. Además, los resultados validan el esquema propuesto en la identificación en línea de eventos novedosos. El análisis comparativo con otros algoritmos de clasificación de última generación reveló que el enfoque de aprendizaje profundo propuesto superó consistentemente a sus contrapartes, destacando su eficacia y rendimiento en el manejo de datos industriales complejos, especialmente en procesos químicos industriales.

Palabras claves: Monitoreo de condiciones, Diagnóstico de fallos, Ciberataques, Aprendizaje profundo, Optimización.

Advanced condition monitoring systems are essential for improving efficiency, safety, and cybersecurity in chemical processes, especially in Industry 4.0. This work proposes an optimized LSTM model for the simultaneous detection and localization of faults and cyberattacks. Deep learning facilitates the robust processing of complex and noisy datasets, improving classification performance even under overlapping fault and attack conditions. The methodology includes an offline phase for data labeling and normalization, as well as optimization of the LSTM model using the Differential Evolution algorithm. This optimization provides the necessary robustness to handle the challenges of industrial data, including noise and external disturbances. During online operation, the system classifies real-time observations under normal operating conditions, faults, or cyberattacks, identifying unknown conditions for expert analysis and system retraining. Evaluated using the Tennessee Eastman (TE) benchmark process, the proposed framework demonstrates superior accuracy and robustness compared to existing methods, particularly in complex scenarios such as faults 9 and 15. Furthermore, the results validate the proposed scheme for the online identification of novel events. Comparative analysis with other state-of-the-art classification algorithms revealed that the proposed deep learning approach consistently outperformed its counterparts, highlighting its effectiveness and performance in handling complex industrial data, especially in chemical processes.

Keywords: Condition monitoring, Fault diagnosis, Cyberattacks, Deep learning, Optimization.

An optimized LSTM model for condition monitoring in chemical processes

Recibido: 10/2025 Aceptado: 12/2025

1. –INTRODUCCIÓN

El concepto de Industria 4.0 representa un paradigma transformador en los entornos industriales y empresariales, impulsado por la integración de tecnologías avanzadas como el Internet Industrial de las Cosas, la computación en la nube y en el borde, el Big Data, la inteligencia artificial y la robótica [1]. Estas tecnologías permiten la digitalización, la conectividad y la automatización de los procesos industriales, transformando las plantas físicas en sistemas ciberfísicos [2]. Si bien esta integración ofrece importantes beneficios, como una mayor eficiencia, calidad del producto y cumplimiento de las normas de seguridad, también plantea nuevos desafíos, especialmente en lo que respeta a fallos relacionados con los procesos y la ciberseguridad [3,4].

En el contexto de los procesos químicos, los fallos relacionados con el proceso y los ciberataques representan riesgos significativos, que pueden derivar en una menor productividad, mayores costos operativos y riesgos para la seguridad [5,6]. Las plantas químicas, con su dinámica compleja y sus estrictos requisitos de seguridad, son particularmente vulnerables a estos riesgos. Por lo tanto, los sistemas avanzados de monitoreo de condición capaces de detectar y localizar tempranamente tanto fallos como ciberataques son esenciales [7]. Sin embargo, lograr este objetivo es complejo debido a las incertidumbres inherentes a las mediciones industriales, como el ruido y las perturbaciones, que dificultan la distinción entre condiciones operativas normales y anormales. Además, los ciberataques a menudo se diseñan para imitar el comportamiento normal de la planta, lo que complica aún más el proceso de detección [8,9].

Los enfoques existentes para el diagnóstico de fallos y la detección de ciberataques en procesos químicos se pueden clasificar, en términos generales, en métodos basados en modelos y métodos basados en datos [10,11]. Los enfoques basados en modelos dependen de un conocimiento profundo de la dinámica de la planta, lo cual suele ser difícil de obtener debido a la complejidad de los sistemas químicos modernos. En cambio, los métodos basados en datos aprovechan los datos históricos para identificar patrones asociados con fallos y ataques, lo que los hace más adaptables a entornos complejos y dinámicos [12,13]. Los avances recientes en inteligencia artificial, en particular el aprendizaje profundo, han demostrado ser muy prometedores para abordar los desafíos de los sistemas de monitoreo de condiciones al procesar eficientemente grandes conjuntos de datos y mejorar la precisión de la clasificación a pesar de las incertidumbres y la superposición de clases de fallos/ataques [14,15]. Sin embargo, el rendimiento de las técnicas de aprendizaje profundo depende en gran medida de la selección adecuada de parámetros, lo que sigue siendo un desafío crítico en las aplicaciones actuales [16].

Una revisión de la literatura reciente revela que el diagnóstico de fallos y la detección de ciberataques se abordan generalmente de forma independiente, a pesar de su objetivo común de garantizar la confiabilidad y seguridad de los sistemas en la industria química [17,18]. Por otro lado, el deterioro progresivo de componentes críticos en sistemas de automatización industrial, como sensores, actuadores y sistemas de bombeo, conduce directamente a la aparición de nuevos fallos. Simultáneamente, la rápida evolución de los entornos de redes industriales, junto con las sofisticadas herramientas y técnicas empleadas por los *hackers*, da lugar a ciberataques nuevos y complejos. En consecuencia, una capacidad crucial para los sistemas de monitoreo de condiciones es la detección de condiciones operativas no reconocidas [19,20].

Este artículo propone un novedoso esquema de monitoreo de condiciones que integra la detección y localización de fallos y ciberataques, abordando ambos desafíos dentro de un marco unificado. Además, la propuesta introduce un novedoso mecanismo en tiempo real para la detección de eventos desconocidos, utilizando máquinas de vectores de soporte de una clase (OCSVM, por sus siglas en inglés) y un algoritmo basado en densidad. Tras la identificación de un evento de este tipo, los especialistas pueden analizar su patrón, actualizar el historial y optimizar el sistema, mejorando así su rendimiento y resiliencia general. Para optimizar los parámetros del modelo LSTM durante la fase de aprendizaje fuera de línea, el marco propuesto incorpora una técnica de optimización bioinspirada: Evolución Diferencial (DE, por sus siglas en inglés).

La estrategia de monitoreo de condiciones propuesta se evalúa mediante el caso de prueba del proceso Tennessee Eastman (TE), una herramienta de simulación ampliamente utilizada en la investigación sobre diagnóstico de fallos y detección de ciberataques [15]. El proceso TE es particularmente relevante para aplicaciones de ingeniería química [21], ya que reproduce las complejidades de las plantas químicas reales. Las simulaciones proporcionan un entorno controlado para replicar diversos escenarios de fallos y ataques, lo que permite realizar pruebas rigurosas y validar la metodología propuesta.

Las principales contribuciones de este artículo se resumen a continuación:

- Una estrategia de monitoreo de condiciones basada en un modelo LSTM que integra la detección y localización de fallos y ciberataques en procesos químicos, fomentando la colaboración entre los equipos de tecnología operativa (TO) y tecnología de la información (TI).

Recibido: 10/2025 Aceptado: 12/2025

- Un mecanismo para detectar fallos y ataques desconocidos, permitiendo un proceso continuo de aprendizaje y adaptación a nuevas amenazas.
- Optimización de parámetros del modelo LSTM que mejora la eficiencia del sistema de monitorización y permite su implementación en tiempo real en entornos industriales.
- Mayor productividad gracias a una toma de decisiones más rápida y precisa, y una reducción de la complejidad computacional gracias a la integración del diagnóstico de fallos y ciberataques y la detección de nuevas anomalías en un único marco de trabajo.

El artículo se estructura de la siguiente forma: La sección 2 describe las características generales de las herramientas computacionales utilizadas en este trabajo, en particular, la Red Recurrente LSTM (*Long-Short-Term Memory*, por su significado en inglés), las Máquinas de Vectores de Soporte de una Clase (OCSVM) y el algoritmo basado en densidad. En esta misma sección se presenta el esquema de monitorización propuesto mediante métodos de aprendizaje profundo. La sección 3 describe el proceso Tennessee Eastman (TE) utilizado para validar la metodología propuesta, el diseño de los experimentos, así como el análisis y la discusión de los resultados. Finalmente, se exponen las conclusiones en la sección 4.

2.- MATERIALES Y MÉTODOS

Esta sección presenta una descripción general de las herramientas computacionales utilizadas en esta investigación, así como se expone el esquema de monitoreo de condiciones diseñado para detectar y localizar fallos y ciberataques.

2.1.- HERRAMIENTAS COMPUTACIONALES

2.1.1- *REDES DE MEMORIA DE LARGO Y CORTO TÉRMINO* (LSTM)

Las redes neuronales recurrentes (RNN) permiten procesar secuencias de vectores a lo largo del tiempo. Uno de los modelos de aprendizaje profundo más potentes para el manejo de datos secuenciales es la red de memoria a corto y largo plazo (LSTM, por sus siglas en inglés), un tipo especializado de RNN. Las LSTM están diseñadas para retener información y descartar detalles irrelevantes en una celda de memoria durante períodos prolongados, lo que les permite capturar y recuperar datos de intervalos de tiempo anteriores durante más tiempo. Esto hace que las LSTM sean especialmente adecuadas para tareas que requieren recordar el contexto a lo largo del tiempo [22].

Una celda LSTM incluye tres puertas: la puerta de olvido ($f_{g(t)}$), la puerta de entrada ($i_{g(t)}$) y la puerta de salida ($o_{g(t)}$), que sirven para gestionar y proteger el estado de la celda. Estas puertas controlan el flujo de información que entra y sale de la celda de memoria, impidiendo que datos irrelevantes sobrescriban la información almacenada. La estructura de una celda LSTM se representa en la Fig. 1. Una celda LSTM opera con $x_{i(t)} \in \mathbb{R}^m$ (entrada), $c_{s(t)} \in \mathbb{R}^n$ (estado de la celda) y $h_{s(t)} \in \mathbb{R}^n$ (estado oculto), donde *m* es el número de características de entrada y *n* es el número de neuronas.

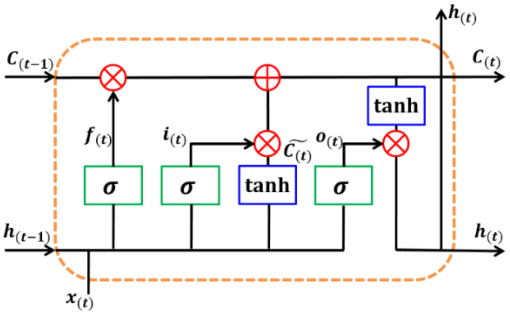

Figura 1

Diagrama de una celda LSTM.

El funcionamiento de una LSTM se describe mediante las Ecuaciones 1-3. $f_{g(t)}$ corresponde a la puerta de olvido en el instante *t*. La función sigmoide (σ) se utiliza para asignar un peso entre 0 y 1. $x_{i(t)}$ representa el vector de entrada en el instante *t*, mientras que $h_{s(t)}$ denota el vector en la capa oculta.

$$f_{g(t)} = \sigma\left(\Psi_{x_i f_g} x_{i(t)} + \Psi_{h_s f_g} h_{s(t-1)} + \Psi_{c_s f_g} c_{s(t-1)} + b_{f_g}\right) \quad (1)$$

$$i_{g(t)} = \sigma\left(\Psi_{x_i i_g} x_{i(t)} + \Psi_{h_s i_g} h_{s(t-1)} + \Psi_{c_s i_g} c_{s(t-1)} + b_{i_g}\right) \quad (2)$$

$$o_{g(t)} = \sigma\left(\Psi_{x_i o_g} x_{i(t)} + \Psi_{h_s o_g} h_{s(t-1)} + \Psi_{c_s o_g} c_{s(t)} + b_{o_g}\right) \quad (3)$$

$c_{s(t)}$ y $h_{s(t)}$ se pueden calcular utilizando las Ecuaciones 4 y 5, como sigue:

$$c_{s(t)} = c_{s(t-1)} f_{s(t)} + tanh\left(\Psi_{x_i c_s} x_{i(t)} + \Psi_{h_s c_s} h_{s(t-1)} + \Psi_{c_s i_g} c_{s(t-1)} + b_{c_s}\right) i_{g(t)} \quad (4)$$

$$h_{s(t)} = tanh\left(c_{s(t)}\right) o_{g(t)} \quad (5)$$

donde $\Psi_{x_i f_g}, \Psi_{x_i i_g}, \Psi_{x_i o_g} \in \mathbb{R}^{nxp}$ y $\Psi_{h_s f_g}, \Psi_{h_s i_g}, \Psi_{h_s o_g} \in \mathbb{R}^{nxn}$ son pesos de la celda, y $b_{f_g}, b_{i_g}, b_{o_g}, b_{c_s} \in \mathbb{R}^n$ son sesgos simbolizados de la celda.

2.1.2- MÁQUINAS DE SOPORTE VECTORIAL DE UNA CLASE (OCSVM)

Las SVM han ganado gran popularidad en tareas de clasificación, ya que utilizan funciones *kernel* para generar límites de decisión no lineales (hiperplanos) mediante la transformación de datos a un espacio de mayor dimensión [23]. La SVM de una clase (OCSVM) es una variante clásica del algoritmo SVM diseñado para la clasificación de una sola clase. De forma similar a las SVM, las OCSVM desarrollan una función de decisión que identifica áreas en el espacio de datos de entrada con una alta densidad de probabilidad. Como se muestra en la Fig. 2, el clasificador OCSVM construye un límite de decisión alrededor de los datos observados durante el entrenamiento y, durante la prueba, identifica los puntos de datos que quedan fuera de este límite como anomalías (clase desconocida mostrada en azul).

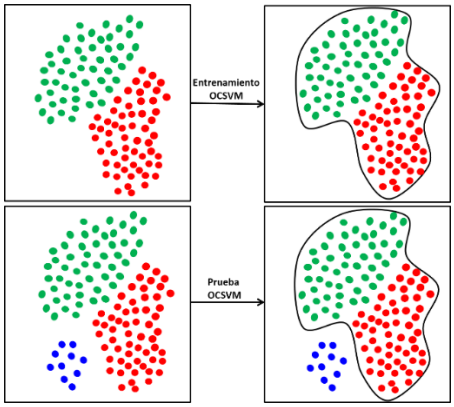

Figura 2

Procedimiento realizado por el algoritmo OCSVM durante el entrenamiento y la prueba.

Un modelo OCSVM se entrena utilizando una variedad de parámetros:

- $\varphi \in (0,1]$: parámetro para el aprendizaje de una sola clase. Se ajusta para controlar el equilibrio entre asegurar que la mayoría de las muestras de entrenamiento pertenezcan a la clase conocida y minimizar los pesos en la función de puntuación.
- Función *kernel*: especifica el tipo de *kernel* que se utilizará; algunos ejemplos son el *kernel* gaussiano o de función de base radial (RBF: utilizado para el aprendizaje de una sola clase), el *kernel* polinómico y el *kernel* lineal. Esto permite que las SVM utilicen una función no lineal para proyectar los datos de entrada a una dimensión superior.

El problema de clasificación de una sola clase busca identificar un hiperplano que aísle una porción específica de los datos de entrenamiento del origen en el espacio de características. Dado que dicho hiperplano no siempre existe en el espacio de características original, se utiliza una función de mapeo **Φ**. El problema se define de la siguiente manera:

$$min_{\mathbf{W},\xi_i,\rho}\left(\frac{1}{2}\|\mathbf{W}\|^2 + \frac{1}{\varphi n}\sum_{i=1}^{n}\xi_i - \rho\right) \quad (6)$$

sujeto a:

$$\langle \mathbf{W}, \mathbf{\Phi}(x_i) \rangle \geq \rho - \xi_i \geq 0, \forall i \tag{7}$$

donde **W** es un vector ortogonal al hiperplano, ξ_i es el *ith* patrón de entrenamiento, *n* es el número total de patrones de entrenamiento, $\xi_i = [\xi_1, \ldots, \xi_n]$ es un vector de variables de holgura utilizadas para penalizar los patrones rechazados, ρ representa el margen, es decir, la distancia del hiperplano desde el origen.

2.1.3- ALGORITMO BASADO EN DENSIDAD

Este algoritmo, basado en el algoritmo de agrupamiento difuso *c-means*, orientado a la densidad (DOFCM, por sus siglas en inglés), busca mitigar la sensibilidad al ruido en la agrupación difusa mediante la detección de valores atípicos antes del proceso de agrupación [24]. Esto se logra creando $g + 1$ clases, de las cuales g son clases y una está destinada al ruido. El algoritmo reconoce los valores atípicos evaluando la densidad del conjunto de datos. En este contexto, para un radio dado, la vecindad de cada punto debe abarcar una cantidad mínima de puntos adicionales. El algoritmo introduce un factor de densidad conocido como pertenencia a la vecindad, que cuantifica la densidad de los objetos en relación con su localidad. La pertenencia local de un punto *i* en el conjunto de datos *X* se puede expresar según la ecuación 8.

$$M^i_{neighborhood} = \frac{\eta^i_{neighborhood}}{\eta_{max}} \tag{8}$$

donde $\eta^i_{neighborhood}$ es el número de puntos en la localidad *i* y η_{max} es la cantidad máxima de valores dentro de la vecindad de cualquiera de los puntos dentro del conjunto. Un punto q es vecino de *i* si q satisface la ecuación 9.

$$q = X | dist(i, q) \leq r_{neighborhood} \tag{9}$$

donde $r_{neighborhood}$ es el radio de vecindad, y $dist(i, q)$ es la distancia que existe entre ambos puntos. El cálculo de $r_{neighborhood}$ se realiza según [25]. La pertenencia a un vecindario de cada punto de *X* se calcula mediante la ecuación 8. El valor α para el umbral se selecciona dentro del rango completo de medidas de pertenencia a un vecindario, en función de la densidad general de elementos en el conjunto de datos. Un punto se clasifica como atípico si su pertenencia a un vecindario es menor que α. Para un *i* específico en *X*, este criterio se expresa según la ecuación 10.

$$\begin{cases} M^i_{neighborhood} < \alpha \text{ entonces } i \text{ valor fuera de rango } (outlier) \\ M^i_{neighborhood} \geq \text{ entonces } i \text{ valor normal } (no\ outlier) \end{cases} \tag{10}$$

2.1.4- ALGORITMO EVOLUCIÓN DIFERENCIAL

Evolución Diferencial (DE) es un algoritmo evolutivo muy conocido que opera con un enfoque poblacional, utilizando métodos bioinspirados como la herencia, la mutación, la selección natural y el cruce. El objetivo principal de DE es generar una población de nuevas soluciones potenciales mediante la modificación de soluciones de la población actual, que ha evolucionado hasta una etapa específica. El procedimiento se guía por los operadores principales: Mutación, Cruce y Selección. En [26] se puede encontrar una explicación detallada de este algoritmo. La Fig. 3 muestra el algoritmo DE y los criterios de parada utilizados son: (1) Número máximo de iteraciones (*Itr_max*) y (2) Valor de la función objetivo.

Entrada: población, factor de escalada, factor de cruce, Itr_max.
Salida: mejor individuo de la población $(F(\hat{\theta}) = Ne)$
Crear una población inicial.
Elija la solución óptima.
for $l = 1$ to $l = Itr\_max$ **do**
    Realizar mutación
    Realizar cruce
    Realizar selección
    Actualizar la solución óptima
    Comprobar el criterio de parada
end for

Figura 3

Algoritmo 1: Evolución Diferencial.

2.2.- ENFOQUE PROPUESTO PARA EL MONITOREO DE CONDICIONES

La Fig. 4 ilustra el enfoque de monitoreo de condiciones diseñado para detectar y localizar fallos y ciberataques. Esta estrategia es adaptable a las industrias de procesos.

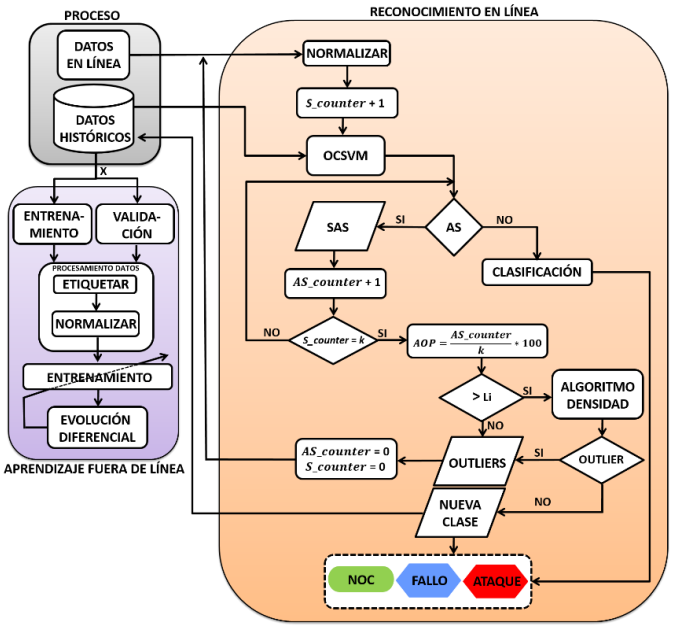

Figura 4

Enfoque de Monitoreo de Condiciones propuesto con detección de clases novedosas.

2.2.1- ETAPA DE ENTRENAMIENTO FUERA DE LÍNEA

La etapa crucial del enfoque de monitorización de condiciones es el entrenamiento. Para desarrollar una red neuronal artificial (RNA) capaz de detectar e identificar fallos y ciberataques en plantas químicas, se deben completar los siguientes pasos: 1) Pre-procesamiento de datos, 2) Diseño de la arquitectura de la RNA, 3) Entrenamiento de la RNA y 4) Validación. La fase de pre-procesamiento comienza etiquetando los datos para cada clase. A continuación, se normaliza el conjunto de datos para evitar que las variables con valores mayores dominen a las de valores menores, garantizando así la retención de toda la información relevante para un entrenamiento eficaz. El conjunto de entrenamiento comprende el 80 % de los datos, con un 70 % destinado al entrenamiento y un 10 % a la validación durante todo el proceso, mientras que el 20 % restante conforma el conjunto de prueba. Los datos de validación se utilizan para evaluar el rendimiento del modelo con los datos de

entrenamiento, ajustando al mismo tiempo sus parámetros. Luego, la red entrenada se prueba con el conjunto de datos de prueba y se pronostica el estado actual del proceso, calculando la precisión para cada clase. Posteriormente, se implementa una etapa para optimizar los parámetros de las redes neuronales utilizadas. En este paso, se estima el número de neuronas (*Ne*) en cada capa oculta *l* de la red mediante la optimización de una función objetivo utilizando un algoritmo de optimización. Esto permite obtener el parámetro *Ne* automáticamente, logrando una arquitectura más eficiente, lo que se traduce en una mejora de la clasificación en la etapa de reconocimiento en línea. La función objetivo utilizada es el Error Cuadrático Medio (MSE, por sus siglas en inglés), como se aprecia en la ecuación 11.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2 \tag{11}$$

En este caso, el MSE corresponde al error de clasificación obtenido, donde N es el número de muestras, Y_i es la precisión ideal (Precisión = 1) y $\hat{Y}_i$ es la precisión obtenida por la red neuronal. Entonces, el problema de optimización se define como:

$$\min\{MSE\} = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2$$

sujeto a:

$$Ne_{min}^1 \leq Ne^1 \leq Ne_{max}^1$$
$$Ne_{min}^2 \leq Ne^2 \leq Ne_{max}^2$$
$$\dots$$
$$Ne_{min}^l \leq Ne^l \leq Ne_{max}^l$$

En numerosas áreas científicas, se han empleado con notable éxito algoritmos inspirados en la biología [27] para abordar desafíos de optimización. Estos algoritmos son eficaces para encontrar la vecindad del óptimo global en la mayoría de los casos, operando dentro de un tiempo de cálculo razonable. En este estudio, se emplea el algoritmo DE para determinar los valores óptimos de los parámetros $Ne^1, Ne^2, \dots, Ne^l$ debido a su sencilla implementación y su excepcional rendimiento. Se llevó a cabo una serie de experimentos para determinar que el rango adecuado era: $32 \leq Ne^l \leq 512$.

2.2.2- ETAPA DE CLASIFICACIÓN EN LÍNEA

Es crucial destacar que, durante la fase de reconocimiento en línea (véase la Fig. 4), el sistema de monitoreo de condiciones propuesto evalúa cada muestra para identificar a qué estado corresponde (NOC: Condición Operativa Normal, Fallo o Ataque, según el entrenamiento). Si se produce un nuevo evento, el esquema de monitoreo identifica una Muestra Anómala (*AS*, por sus siglas en inglés). Durante esta etapa, la propuesta permite la detección de nuevas clases. Un grupo de especialistas establece una ventana temporal con *k* observaciones y un criterio denominado *Li*. El valor de *k*, determinado por los atributos del procedimiento, representa la cantidad de tiempos de muestreo que estos expertos consideran suficientes para evaluar la posible presencia de un nuevo patrón. *Li* indica el porcentaje de valores observados que los especialistas determinan para evaluar si el conjunto de observaciones identificadas como anomalías (denominado Conjunto de Muestras Anómalas (*SAS*, por sus siglas en inglés)) en *k* instancias de muestreo debe examinarse para determinar si representa una nueva clasificación, lo que indicaría un posible nuevo evento.

Este trabajo propone un nuevo algoritmo para la detección en línea de clases desconocidas. En esta fase, el algoritmo OCSVM evalúa cada nueva observación del proceso para determinar si corresponde a un estado operativo conocido. Para ello, el clasificador OCSVM construye un límite de decisión alrededor de los datos observados durante el entrenamiento y, durante la fase en línea, identifica los puntos de datos fuera de este límite como *SAS*. Si la muestra corresponde a un estado operativo conocido, la herramienta basada en aprendizaje profundo la categoriza en consecuencia. Sin embargo, si la muestra se marca como *AS*, se guarda y se incrementa una variable contador que representa las muestras anómalas. Este proceso se repite hasta completar el intervalo de tiempo de *k* observaciones.

Tras clasificar las *k* observaciones, se calcula el porcentaje de observaciones identificadas como anómalas (*AOP = AS*100/k*). Si el *AOP* no supera el umbral *Li*, las muestras no se consideran indicativas de un nuevo evento y el contador *AS* se reinicia para el siguiente ciclo. Sin embargo, si el *AOP* supera el *Li*, las observaciones *AS* se evalúan con mayor detalle para determinar si representan una nueva clase, lo que podría indicar un nuevo fallo o ataque, o si simplemente son *outliers*. Para analizar las observaciones anómalas, se emplea un algoritmo basado en densidad. Este algoritmo opera bajo el principio de que los valores atípicos son puntos de datos dispersos con baja densidad y no forman grupos definidos. Por el contrario, la aparición de un nuevo patrón, como un fallo o un ataque, generalmente provoca que los datos se agrupen estrechamente, lo que aumenta la

densidad y da lugar a la formación de un estado distinto. El algoritmo analiza las muestras atípicas para determinar, según la densidad de los puntos de datos, si son verdaderos valores atípicos o si indican un nuevo patrón de clasificación. Si las muestras anómalas se identifican como un nuevo evento, los especialistas deben determinar si el patrón corresponde a un fallo o un ataque. Una vez detectado y clasificado el nuevo evento, y si se confirma que el patrón corresponde a un nuevo tipo de fallo o ataque, se registra en la base de datos de entrenamiento. Posteriormente, se reentrena el algoritmo de clasificación y se itera sistemáticamente el proceso de monitoreo en línea, incluyendo el procedimiento de detección de nuevos fallos/ataques. Esta fase en línea sirve como mecanismo para la detección automatizada de nuevas amenazas, incorporando capacidades de aprendizaje continuo a un sistema de gestión de ciberataques y seguridad.

3.- DISEÑO DE EXPERIMENTOS Y ANÁLISIS DE LOS RESULTADOS

3.1.- PROCESO DE PRUEBA: TENNESSEE EASTMAN (TE)

El proceso de referencia Tennessee Eastman (TE) se utiliza para evaluar el esquema de monitoreo de condiciones propuesto [9]. La planta consta de cinco subprocesos interconectados (véase la Fig. 5): reactor, condensador, separador vapor-líquido, separador de producto y compresor.

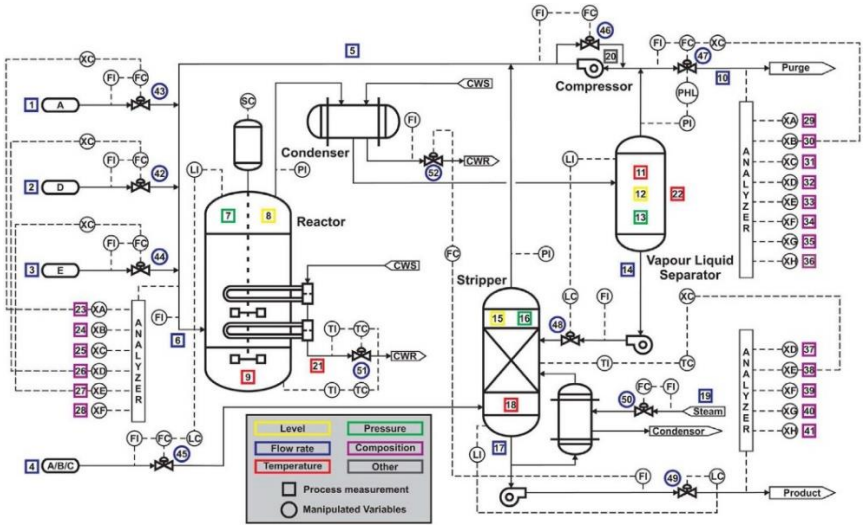

Figura 5

Esquema del proceso TE.

En el proceso TE intervienen 52 variables y un conjunto de datos relacionados tanto con las Condiciones Normales de Operación como con 20 fallos diferentes. Además, deben tenerse en cuenta las variables de entrada en las mediciones, ya que podrían indicar la presencia de un ciberataque, especialmente cuando los hackers intentan ocultarlo en el modo de Condiciones Normales de Operación. Cada conjunto de muestras abarca 48 horas, con fallos introducidos tras 8 horas de simulación, lo que añade ruido a las mediciones (Fuente: http://web.mit.edu/braatzgroup/-TE process.zip). La Tabla 1 enumera los 20 fallos utilizados para analizar y evaluar el enfoque propuesto (se sabe que F3, F9 y F15 son difíciles de clasificar correctamente debido a su gran parecido con el NOC). En la fase de entrenamiento, se utilizan 500 observaciones de cada fallo y del NOC, mientras que durante la fase de prueba se utilizan 960 observaciones. Los ataques utilizados se detallan en la Tabla 2, donde el *hacker* posee un conocimiento completo del proceso y puede manipular las mediciones de los sensores en cualquier momento [9]. Se utilizó Simulink/MATLAB para desarrollar los experimentos. Al igual que con los escenarios de fallos, se emplearon 500 muestras. Los conjuntos de datos de entrenamiento y prueba constan de 500 simulaciones cada uno para cada fallo/ataque y la Condición Operativa Normal (NOC), con la siguiente estructura: 350 para entrenamiento (70 %), 50 para validación (10 %) y 100 para prueba (20 %). Por lo tanto, el tamaño final de los conjuntos de datos es: (1) Entrenamiento: 175 000. (2) Validación: 25 000. (3) Prueba: 96 000.

Tabla 1
Fallos simulados en el proceso TE.

Fallo	Descripción	Tipo
F1	Relación de alimentación A/C, composición de B constante (Corriente 4)	Escalón
F2	Composición de B, relación A/C constante (Corriente 4)	Escalón
F3	Temperatura de alimentación D (Corriente 2)	Escalón
F4	Temperatura de entrada del agua de enfriamiento del reactor	Escalón
F5	Temperatura de entrada del agua de enfriamiento del condensador (Corriente 2)	Escalón
F6	Pérdida de alimentación A (Corriente 1)	Escalón
F7	Pérdida de presión en el cabezal C (disminución de disponibilidad) (Corriente 4)	Escalón
F8	Temperatura de entrada del agua de enfriamiento del reactor	Variación aleatoria
F9	Temperatura de alimentación D (Corriente 2)	Variación aleatoria
F10	Temperatura de alimentación C (Corriente 4)	Variación aleatoria
F11	Temperatura de entrada del agua de enfriamiento del reactor	Variación aleatoria
F12	Temperatura de entrada del agua de enfriamiento del condensador	Variación aleatoria
F13	Cinética de reacción	Deriva lenta
F14	Agua de enfriamiento del reactor	Válvula atascada
F15	Válvula de agua de enfriamiento del condensador	Válvula atascada
F16	Desviaciones en la transferencia de calor dentro del separador	Variación aleatoria
F17	Desviaciones en la transferencia de calor dentro del reactor	Variación aleatoria
F18	Desviaciones en la transferencia de calor dentro del condensador	Variación aleatoria
F19	Válvula de recirculación del compresor, separador inferior y válvula de vapor del separador	Válvula atascada
F20	Desconocido	Variación aleatoria

Tabla 2
Ataques simulados en el proceso TE.

Ataque	Sensor atacado	Variables de síntoma	Descripción	Efecto
A1	XME(1) [+2.35]	XME(1), XME(7), XME(8), XMV(3)	En 3 horas, el valor aumentó en 2.35 kscmh	Alta presión en el reactor o parada por bajo nivel en el stripper
A2	XME(14) [+7]	XME(12), XME(14), XME(15), XMV(7), XMV(8)	En 2.88 horas el valor aumentó en 7 m^3/h	Parada por alto nivel en el stripper
A3	XME(14) [–7]	XME(12), XME(14), XME(15), XMV(7), XMV(8)	En 2.02 horas el valor se redujo en 7 m^3/h	Parada por bajo nivel en el separador
A4	XME(14) [22.9]	XME(12), XME(15), XMV(7)	Se fijó el valor en 22.9 m^3/h durante 1.9 horas	Parada por bajo nivel en el separador

XME(1):Alimentación A, XME(15): Nivel del stripper, XME(7): Presión del reactor, XMV(3): Flujo de alimentación A, XME(12): Nivel del separador, XMV(7): Flujo del separador, XME(14): Descarga del separador, XMV(8): Flujo del stripper.

3.2.- DISEÑO DE EXPERIMENTOS

3.2.1- ETAPA DE ENTRENAMIENTO FUERA DE LÍNEA

En esta etapa, es necesario normalizar el conjunto de datos para que todos se encuentren en un rango de valores similar (entre 0 y 1). Otra etapa en el procesamiento de datos es el etiquetado, dado que los modelos aplicados son de aprendizaje supervisado. En este problema hay 23 clases, por lo que el vector de etiquetas tendrá dimensión 23 (NOC, 19 Fallos y 3 Ataques). Los patrones de dos clases (Fallo 20 y Ataque 4) no se utilizarán para entrenar las redes neuronales, estos permitirán validar el reconocimiento de nuevos eventos en la etapa en línea.

Los pasos para configurar la red LSTM son: (1) Configurar la capa de entrada para que coincida con el tamaño de las señales de entrada (52). (2) Se definen tres capas ocultas LSTM, cada una con Ne^1, Ne^2 y Ne^3 unidades, respectivamente (estos parámetros se optimizarán mediante el algoritmo DE). (3) Para evitar el sobreajuste, se insertan tres capas de *dropout* entre las capas LSTM. Durante cada iteración de entrenamiento, cada unidad y sus conexiones se retiran temporalmente de la red con una probabilidad $Pr_{dropout}$ (en este caso, $Pr_{dropout} = 0.5$). El *dropout* se aplica únicamente durante el entrenamiento y se ignora durante la prueba. (4) Por último, para la clasificación, agregue una capa totalmente conectada cuyo tamaño coincida con el número de clases de salida (23 en total: NOC, 3 ataques y 19 fallos). A continuación, se incorpora una capa *softmax* para calcular las puntuaciones de probabilidad de cada clase en el escenario multiclase. Concluya con una capa de clasificación que genere la salida final: fallo, ataque o NOC.

Para estimar el parámetro *Ne*, se eligió el algoritmo DE debido a sus ventajas, como su estructura sencilla, su rápida ejecución y su gran robustez [28]. En la implementación, se consideraron tres estrategias: a) aquellas con mayor tendencia a la

diversificación (exploración), b) aquellas con mayor tendencia a la intensificación (explotación) y c) aquellas con capacidades similares de diversificación e intensificación. Tras aplicar la prueba estadística no paramétrica de Friedman, se concluyó que no existían diferencias significativas entre las tres estrategias. Por este motivo, se seleccionó la tercera estrategia. Los parámetros de control en DE son el tamaño de la población *Z*, la constante de cruce C_R y el factor de escala F_S. Los valores de los parámetros fueron: $C_R = 0.5, F_S = 0.1$, $Z = 10$, $Itr\_max = 100$ y $MSE \leq 0.0001$. En este caso, el espacio de búsqueda para cada red neuronal utilizada es: $32 \leq Ne^l \leq 512$. Para el algoritmo OCSVM, los valores de los parámetros utilizados fueron: $\varphi = 0.5$ y $\sigma = 10$.

3.2.2- ETAPA DE CLASIFICACIÓN EN LÍNEA

En esta etapa, se llevaron a cabo cinco experimentos para validar la etapa de clasificación en línea.

- En el Experimento 1, se tomaron 100 observaciones: 80 correspondientes a la clase NOC y 20 *outliers* que representan posibles perturbaciones que afectan el proceso industrial. Este experimento tuvo como objetivo analizar la efectividad de la propuesta para clasificar satisfactoriamente las muestras de las clases conocidas (en particular, la Condición Normal de Operación) y la capacidad de detectar múltiples muestras como clases desconocidas que constituyen valores atípicos, para luego decidir no considerar ninguna acción adicional si se cumple la condición $AOP < Li$.
- En el Experimento 2, se consideraron 80 muestras del Fallo F3 y 20 muestras como *outliers*. El objetivo de este experimento fue evaluar el algoritmo propuesto para clasificar correctamente las clases conocidas (en este caso, se seleccionó el Fallo F3 debido a su similitud con NOC y, por lo tanto, a la dificultad de clasificarla) y analizar la capacidad de detectar valores atípicos (cuando se cumple la condición $AOP < Li$).
- En el Experimento 3, se consideraron 20 muestras pertenecientes al Fallo F15 (este fallo se seleccionó por su alta dificultad de clasificación debido a su patrón similar al de NOC) y 80 valores atípicos. El objetivo de este experimento es evaluar el funcionamiento del algoritmo en línea para detectar varios valores atípicos, como los de clase desconocida. Tras verificar la condición $AOP > Li$, se puede evaluar la capacidad de clasificar estas muestras como valores atípicos, dado que no constituyen una categoría separada debido a su baja densidad de datos.
- En el Experimento 4, se consideraron 80 muestras del Fallo F20 (Nueva Clase) y 20 *outliers*. Este experimento permite verificar el rendimiento del algoritmo en línea para clasificar inicialmente todas las observaciones como de clase desconocida y, posteriormente, clasificar los *outliers* y detectar un nuevo evento derivado de la densidad de datos.
- En el Experimento 5, se consideraron 80 muestras del ataque A4 (Nueva Clase) y 20 valores atípicos. El objetivo es similar al del experimento 4, excepto que en este caso se evalúa el rendimiento del algoritmo de clasificación en línea para detectar un nuevo ciberataque.

3.3.- ANÁLISIS DE LOS RESULTADOS

3.3.1- ETAPA DE ENTRENAMIENTO FUERA DE LÍNEA

El valor de la función objetivo (MSE) para el algoritmo LSTM-DE se muestra en la Fig. 6. Se observa que, el valor converge rápidamente, lo cual representa otra ventaja de DE como algoritmo de optimización. Los valores estimados de los parámetros (*Ne*) fueron: $Ne^1 = 128$, $Ne^2 = 64$ *y* $Ne^3 = 32$. La Fig. 7 muestra la evolución de los parámetros mejores estimados en cada iteración. Se observa que el algoritmo DE presenta una excelente capacidad de explotación, ya que los parámetros convergen rápidamente.

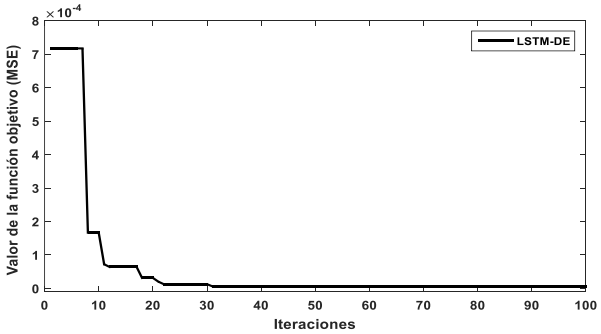

Figura 6
Valor de la función objetivo (MSE).

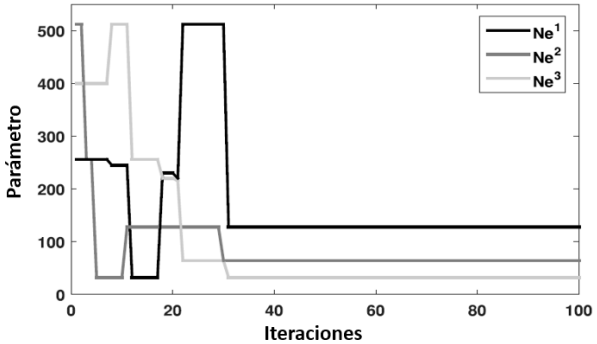

Figura 7
Evolución de los parámetros.

La Fig. 8 muestra el error de entrenamiento obtenido por la red LSTM utilizando los parámetros estimados por el algoritmo ED. Esto demuestra que el modelo LSTM funciona correctamente sin sobreajuste tras 467 épocas de entrenamiento. Las especificaciones del ordenador utilizado para realizar los experimentos fueron: Intel Core i7-6600U de 2,6 a 2,81 GHz, memoria: 16 GB DDR3L. El tiempo medio de ejecución del algoritmo LSTM en la fase de aprendizaje fuera de línea fue de 954 minutos y 37 segundos. En el caso del algoritmo OCSVM, el tiempo medio de ejecución fue de 1 minuto y 15 segundos.

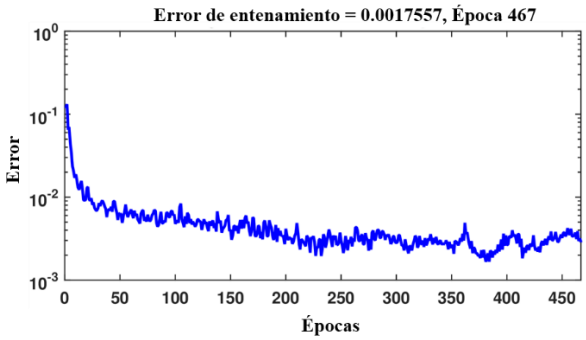

Figura 8

Error de entrenamiento.

3.3.2- ETAPA DE CLASIFICACIÓN EN LÍNEA

Analizar la calidad y el rendimiento de un sistema de clasificación es crucial. Una herramienta comúnmente utilizada para este análisis es la matriz de confusión, que ayuda a evaluar el rendimiento de los algoritmos de clasificación. Los valores en la diagonal principal indican observaciones clasificadas correctamente, mientras que los valores fuera de la diagonal reflejan casos de clasificación errónea. Para detectar rápidamente una nueva clase, se consideró evaluar 100 muestras. Esto implica considerar una ventana de tiempo de tamaño k = 100 (100 segundos). Se decidió utilizar un umbral de decisión de Li = 60% para establecer un nivel adecuado de mayoría de muestras clasificadas como de clase desconocida. Es importante destacar que los criterios para seleccionar estos parámetros dependen del tipo de proceso y de la opinión de los expertos. Las Tablas 3-5 muestran las matrices de confusión para los cinco experimentos descritos anteriormente. Se consideraron los estados: NOC, F1, F2,..., F19, A1, A2, A3, la clase Nueva (NC) y la clase de *Outliers* (O). La diagonal principal de la matriz de confusión indica el número de muestras identificadas o clasificadas correctamente, mientras que los valores fuera de la diagonal reflejan las clasificaciones erróneas. La última fila muestra la precisión promedio (AVE) en todas las clases.

La Tabla 3 muestra los resultados obtenidos en la etapa de clasificación en línea. En todos los casos, se obtuvieron resultados satisfactorios gracias a una clasificación precisa de la clase NOC, F3 y los valores atípicos. En el experimento 1, el algoritmo OCSVM clasificó las 20 muestras como *SAS*, y posteriormente $AOP < Li$, por lo que no se evaluó la densidad de *SAS*. En el experimento 2, el algoritmo OCSVM clasifica 19 observaciones como *SAS*, luego $AOP < Li$ entonces no se evaluó la densidad de *SAS*. El único valor atípico identificado como clase conocida se analiza mediante el algoritmo LSTM y se clasifica en la clase del Fallo F16.

Tabla 3

Matriz de Confusión para clasificación en línea: Experimento 1 y 2 (NOC: 80, F3: 80, O: 20).

	Experimento 1			**Experimento 2**		
	NOC	SAS	AVE	F3	SAS	AVE
NOC	**80**	0		0	0	
F1-F2	0	0		0	0	
F3	0	0		**80**	0	
F4-F15	0	0		0	0	
F16	0	0		0	1	
F17-F19	0	0		0	0	
A1-A3	0	0		0	0	
O	0	**20**		0	**19**	
TA (%)	100	100	**100**	100	95	**99**

Para el experimento 3, la Tabla 4 presenta los resultados obtenidos. Se evidencia que el algoritmo en línea logra resultados satisfactorios en la identificación de 20 muestras del Fallo F15 y los 80 *outliers*. En este caso, el algoritmo OCSVM clasifica 78 observaciones como *SAS*, por lo que *AOP > Li*. Posteriormente, se evaluó la densidad de *SAS*. Los 2 *outliers* identificados como de clase conocida se analizan mediante el modelo LSTM y se clasifican en las clases de los Fallos F15 y F16. La Tabla 5 muestra los resultados de la evaluación de las muestras identificadas como *SAS*. En este caso, el algoritmo basado en densidad debe determinar cuáles de las *SAS* se clasifican como un nuevo evento o como valores atípicos. Cabe destacar que, en este análisis, el algoritmo basado en densidad se emplea para detectar una única clase que representa la nueva clase, mientras que las *SAS* incluyen los valores atípicos. Estos resultados deben ser evaluados por expertos para determinar si existe un nuevo evento que represente un nuevo patrón de ataque o fallo. Las observaciones identificadas como un nuevo evento son relativamente pequeñas en comparación con el número total de muestras analizadas y, por lo tanto, no se consideran como tal. En consecuencia, debido a esta decisión, se eliminan y se reinician los contadores *AS* y *S*.

Tabla 4
Matriz de Confusión para clasificación en línea: Experimento 3 (F15: 20, O: 80).

	Experimento 3		
	F15	SAS	AVE
NOC	1	0	
F1-F14	0	0	
F15	**19**	1	
F16	0	1	
F17-F19	0	0	
A1-A3	0	0	
O	0	**78**	
TA (%)	95	97.5	**97**

Tabla 5
Matriz de Confusión para clasificación en línea: Experimento 3 (Algoritmo basado en Densidad, SAS: 78).

	Experimento 3
	SAS
NOC	2
O	**76**
TA (%)	**97.44**

La Tabla 6 presenta los resultados de la evaluación y clasificación de 80 muestras pertenecientes a una nueva clase (Fallo F20/Ataque A4) y 20 observaciones atípicas en 100 periodos de muestreo. En el experimento 4, el algoritmo OCSVM clasifica 97 observaciones como SAS, por lo que AOP > Li y el conjunto de valores atípicos deben analizarse. Las observaciones AS identificadas como una clase conocida se analizan mediante el modelo LSTM y se clasifican en la clase del Fallo F15. En el experimento 5, el algoritmo OCSVM clasifica 99 muestras como SAS. La AS identificada como de clase conocida se analiza y se clasifica en la clase NOC. La Tabla 7 muestra los resultados tras la implementación del algoritmo basado en densidad para clasificar los *AS* como valores atípicos o como un nuevo evento. La presencia de una nueva clase (Fallo F20 o Ataque A4), es claramente evidente. Una vez identificada la nueva clase, los expertos deben caracterizar este patrón y actualizar la base de datos histórica para reentrenar los algoritmos del sistema de monitoreo de condición, incorporando así el nuevo patrón.

Tabla 6
Matriz de Confusión para clasificación en línea: Experimento 4 y 5 (NC (F20): 80, NC (A4): 80, O: 20).

	Experimento 4	**Experimento 5**
	AS	AS
NOC	0	1
F1-F14	0	0
F15	3	0
F16-F19	0	0
A1-A3	0	0
O	**97**	**99**
TA (%)	**97**	**99**

Tabla 7
Matriz de Confusión para clasificación en línea: Experimento 4 y 5.

	Experimento 4			**Experimento 5**		
	NC	O	AVE	NC	O	AVE
NC	**75**	1		**78**	0	
O	3	**18**		1	**20**	
TA (%)	96.15	94.74	**95.88**	98.73	100	**98.99**
	NC(F20): 78, O:19			NC(A4): 79, O:20		

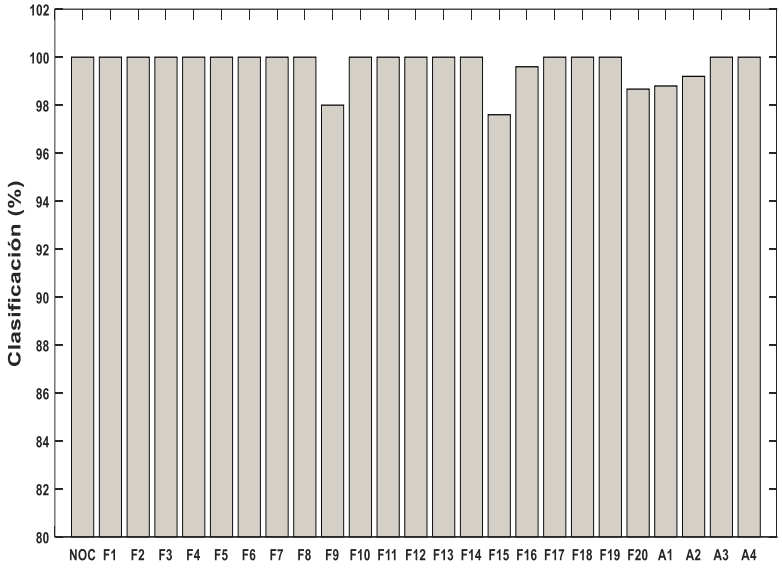

Figura 9

Clasificación para todas las clases en el proceso TE.

La Fig. 9 muestra la clasificación una vez que los nuevos patrones (Fallo F20 y Ataque A4) se identifican y actualizan en la base de datos de entrenamiento. Como se puede observar, se obtienen excelentes resultados, destacando el buen rendimiento para los Fallos F9 y F15. Estos fallos son conocidos por su dificultad para ser clasificados correctamente debido a su gran similitud con el NOC.

3.3.3- COMPARACIÓN CON HERRAMIENTAS SIMILARES

Como se muestra en la Tabla 8, los resultados de clasificación de los fallos usando el algoritmo propuesto se contrastaron con los obtenidos con otras redes neuronales comparables [29,30]. Las arquitecturas utilizadas en la comparación son: CNN-LSTM, Multiscale Residual Jagged Dilated Convolutional (MRJDCNN), MRJDCNN-LSTM, Dilated Convolution Bi-directional Gated Recurrent Unit (DCNN-BiGRU) y Dilated Convolution-stacked Bidirectional gated Recurrent Unit (HLFFDCNN-BiGRU). Los resultados muestran que todas las clasificaciones de fallos son altas, siendo los Fallos F3, F9 y F15 las más difíciles de identificar debido a su similitud con el funcionamiento normal del proceso. Además, se observa que la precisión del esquema propuesto es mayor que la de los demás modelos. Por lo tanto, se confirma que el rendimiento del esquema propuesto en este trabajo es superior. Al analizar los fallos complejos F3, F9 y F15, se puede observar en la Fig. 10 un histograma de las puntuaciones F1 (*F1 score*) para cada arquitectura utilizada. En [30] el Fallo F15 se excluye de esta comparación.

Además, la Tabla 9 presenta una comparación con las técnicas utilizadas en [9] para clasificar los ataques analizados, incluyendo Regresión Logística (LR, por sus siglas en inglés), Regresión Logística con Validación Cruzada (LRCV, por sus siglas en inglés), Red Neuronal (NN, por sus siglas en inglés) y Bosque Aleatorio (RF, por sus siglas en inglés). Los números resaltados identifican los mejores resultados, lo que certifica la superioridad de la propuesta presentada.

Tabla 8
Comparación de los resultados de clasificación (en %) con modelos recientes de aprendizaje profundo.

Fallo	CNN-LSTM [29]	MRJDCNN [29]	MRJDCNN-LSTM [29]	DCNN-BiGRU [30]	HILFFDCNN-BiGRU [30]	Propuesta
F1	98.46	**100**	**100**	**100**	**100**	**100**
F2	**100**	98.91	**100**	**100**	99.58	**100**
F3	80.31	85.33	91.87	78.33	93.87	**100**
F4	96.96	93.56	96.09	95.26	96.30	**100**
F5	90.16	**100**	**100**	98.00	99.20	**100**
F6	**100**	**100**	**100**	**100**	**100**	**100**
F7	**100**	99.65	**100**	**100**	**100**	**100**
F8	95.39	96.78	98.01	97.59	**100**	**100**
F9	74.30	86.25	92.71	82.17	92.96	**98.00**
F10	92.86	95.64	96.31	96.43	98.05	**100**
F11	88.10	92.56	95.73	95.48	95.98	**100**
F12	94.29	98.11	97.05	99.20	98.79	**100**
F13	93.79	97.85	98.67	99.23	98.10	**100**
F14	96.95	96.04	98.23	98.79	99.23	**100**
F15	87.22	85.37	89.93	79.77	96.19	**97.60**
F16	89.92	89.24	94.87	92.78	97.25	**99.60**
F17	91.40	93.31	96.67	95.97	97.70	**100**
F18	96.21	95.27	97.31	98.83	99.63	**100**
F19	86.09	87.96	91.30	85.48	94.98	**100**
F20	95.93	92.09	97.10	91.67	97.56	**98.67**
AVE	92.42	94.20	96.59	94.25	97.77	**99.69**

Los números en negrita identifican los mejores resultados.

Tabla 9
Comparación de los resultados de clasificación (%) de los ataques con otras técnicas.

Ataque	LR	LRCV	NN	RF	Propuesta
A1	63.30	63.30	90.10	95.90	**98.80**
A2	86.50	86.50	97.00	**100**	99.20
A3	76.00	74.00	90.20	93.80	**100**
A4	68.40	72.30	71.90	75.00	**100**
AVE	73.55	74.03	87.30	91.18	**99.50**

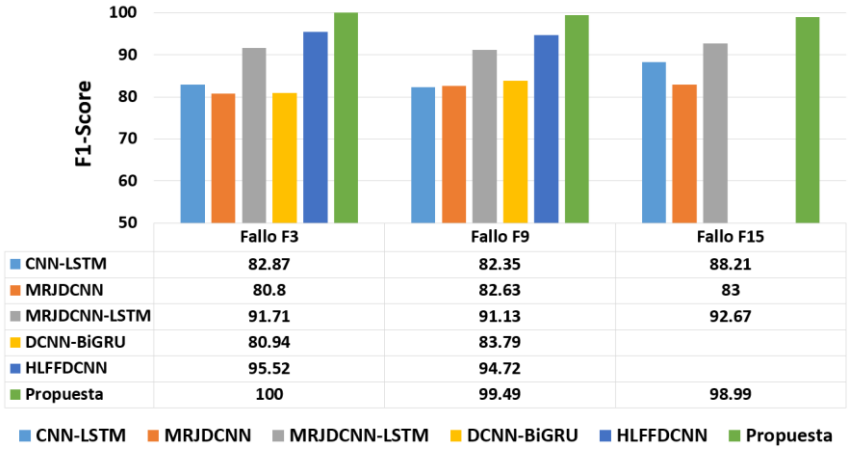

Figura 10

Valores de *F1 score* con respecto a los Fallos F3, F9 y F15.

4.- CONCLUSIONES

Este artículo presenta una estrategia robusta e innovadora de monitorización de condiciones para plantas industriales, que aprovecha algoritmos de aprendizaje profundo para abordar los desafíos de la detección y localización integradas de fallos y ciberataques, así como la detección de eventos desconocidos, en un único marco de trabajo. El enfoque propuesto logra la integración de tareas tradicionalmente separadas, mejorando así la colaboración entre los equipos de Tecnología Operativa (TO) y Tecnología de la Información (TI). Esta integración no solo mejora la fiabilidad y la seguridad del sistema, sino que también reduce la complejidad computacional, convirtiéndola en una solución práctica para entornos industriales modernos.

El uso del aprendizaje profundo permite el procesamiento eficaz de conjuntos de datos grandes y con ruido, mejorando la precisión de la clasificación incluso en presencia de categorías de fallos y ataques superpuestas. Durante la fase de entrenamiento fuera de línea, se empleó el algoritmo de Evolución Diferencial para optimizar los parámetros de la red LSTM. Esta optimización garantiza que los modelos sean idóneos para gestionar las complejidades de los datos industriales, incluyendo el ruido y las perturbaciones externas.

En la fase de reconocimiento en línea, el sistema analiza observaciones en tiempo real para clasificarlas en Condiciones Normales de Operación, fallos o ciberataques. Además, se introdujo una metodología novedosa para detectar y caracterizar eventos desconocidos, como nuevos tipos de fallos o ataques. Cuando se identifican dichos eventos, se añaden a la base de datos histórica y se reentrenan los clasificadores, lo que garantiza la adaptabilidad del sistema a las amenazas emergentes.

La estrategia propuesta se validó utilizando el proceso de referencia Tennessee Eastman (TE), un estándar ampliamente reconocido para evaluar métodos de diagnóstico de fallos y detección de ciberataques. Los resultados demostraron la robustez y el alto rendimiento del sistema, especialmente en escenarios complejos como los Fallos 9 y 15. El análisis comparativo con otros algoritmos de clasificación de última generación reveló que el enfoque propuesto superó consistentemente a sus contrapartes, destacando su eficacia en el manejo de datos industriales complejos, alcanzando una precisión superior al 95%. Cabe destacar también la capacidad del esquema propuesto para identificar nuevos eventos en línea, como lo demuestran los experimentos realizados en la subsección 3.3.2.

Las futuras líneas de investigación incluyen la exploración de la integración de técnicas de aprendizaje no supervisado y el estudio de su aplicabilidad en sistemas embebidos en tiempo real. Esto permitirá identificar patrones en datos no etiquetados, logrando un mejor rendimiento en la detección y localización de fallos y ciberataques. Este trabajo contribuye al avance de la inteligencia computacional en aplicaciones de ingeniería, ofreciendo una solución escalable y eficiente a los retos de la Industria 4.0.

REFERENCIAS

1. Macas, M., Wu, C., Fuertes, W., A survey on deep learning for cybersecurity: progress, challenges, and opportunities. Computer Network. 2022; (212):1–33.
2. Suárez Concepción, F., Piñero Aguilar, R., Prieto Moreno, A.S., Alfonso Cordoví, A., Carbó Castro, J.C., Llanes-Santiago, O. Metodología para la automatización de procesos tecnológicos en la industria farmacéutica cubana. Revista Ingeniería Industrial. 2022; 43(1):1-14.
3. Lv, H., Chen, J., Pan, T., Zhang, T., Feng, Y., Liu, S.Attention mechanism in intelli- gent fault diagnosis of machinery: a review of technique and application. Measurement. 2022; 199:111594.
4. Azzam, M., Pasquale, L., Provan, G., Nuseibeh, B., 2023. Forensic readiness of industrial control systems under stealthy attacks. Computer & Security. 2023; 125:1–10.
5. Alanazi, M., Mahmood, A., Morshed, M., Scada vulnerabilities and attacks: a review of the state of the art and open issues. Computer & Security. 2023; 125:1–29.
6. Gómez, A. L.P., Maimó, L.F., Celdrán, A.H., Clemente, F. J.G., Vaasi, Crafting valid and abnormal adversarial samples for anomaly detection systems in industrial scenarios. Journal of Information Security and Applications. 2023; 79:103647.
7. Anthi, E., Williams, L., Rhode, M., Burnap, P., Wedgbury, A., Adversarial attacks on machine learning cybersecurity defences in industrial control systems. Journal of Information Security and Applications. 2021; 58:102717.
8. Zang, P., Wen, G., Dong, S., Lin, H., Huang, X., Tian, X., Chen, X., A novel mul- tiscale lightweight fault diagnosis model based on the idea of adversarial learning. Neurocomputing. 2018; 275:1674–1683.
9. Kravchik, M., Demetrio, L., Biggio, L., Shabtai, A., Practical evaluation of poisoning attacks on online anomaly detectors in industrial control system. Computer & Security. 2022; 301:120699,
10. Rao, S., Wang, J., 2024. A comprehensive fault detection and diagnosis method for chemical-processes. Chemical Engineering Science. 2024; 300:120565.

11. Quiñones-Grueiro, M., Prieto-Moreno, A.S., Llanes-Santiago, O. Modeling and monitoring for transitions based on local kernel density estimation and process pattern construction .Industrial & Engineering Chemistry Research. 2016; 55(3):692-702.
12. Hadroug, N., Hafaifa, A., Alili, B., Iratni, A., Chen, X., Fuzzy diagnostic strategy implementation for gas turbine vibrations faults detection: towards a characterization of symptom fault correlations. Journal of Vibration Engineering & Technologies. 2022; 10:225–251.
13. Lundgren, A., Jung, D., Data-driven fault diagnosis analysis and open-set classification of time-series data. Control Engineering Practice. 2022; 121:105006.
14. Zheng, S., Li, S., Wang, Y., A novel BDPCA-SMLSTM algorithm for fault diagnosis of industrial process. Chemical Engineering Science. 2025; 305:121172.
15. Khan, N.A., Sulaiman, M., Lu, B., Predictive insights into nonlinear nanofluid flow in rotating systems: a machine learning approach. Engineering with Computers. 2024. https://doi.org/10.1007/s00366-024-01993-1.
16. Rodríguez-Ramos, A., Bernal de Lázaro, J.M., Silva Neto, A.J., Llanes-Santiago, O., Fault detection using kernel computational intelligence algorithm, computational intelligence. In: Optimization and Inverse Problems with Applications in Engineering. 2019; 63–75.
17. Polat, H., Turkoglu, M., Polat, O., Sengur, A., A novel approach for accurate detection of the ddos attacks in sdn-based scada systems based on Deep recurrent neural networks, Expert Systems With Applications. 2022; 197:116748.
18. Shaukat, K., Luo, S., Varadharajan, V., A novel method for improving the robustness of deep learning-based malware detectors against adversarial attacks, Engineering Applications of Artificial Intelligence. 2022; 116:105461.
19. Huang, H., Li, T., Ding, Y., Li, B., Liu, A., An artificial immunity based intrusion detection system for unknown cyberattacks. Applied Soft Computing. 2023; 148:110875.
20. Rao, S., Wang, J., A, Expert label for explainable fault diagnosis and for unknown fault generalization, Chemical Engineering Science. 2025; 301:120699.
21. Cao, Y., Li, P., Deng, X., Chemical process fault diagnosis based on bi-level dynamic IndRNN. Chemical Engineering Science. 2025; 307:121335.
22. Lindemann, B., Maschler, B., Sahlab, N., Weyrich, M. A survey on anomaly detection for technical systems using lstm networks, Computers in Industry. 2021; 131:1--11.
23. Cuong-Le, T., Nguyen, T.N., Khatir, S., Nguyen, P.T., Mirjalili, S., Nguyen, K.D., An efcient approach for damage identifcation based on improved machine learning using pso-svm. Engineering with Computers. 2022; 38:3069–3084.
24. Yadav, H., Singh, J., Gosain, A., Experimental analysis of fuzzy clustering techniques. Procedia Computer Science. 2023; 218:959 – 968.
25. Ester, M., Kriegel, H., Sander, J., Xu, X., 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the 2nd ACM SIGKDD, 1996; 226–231.
26. Rodríguez-Ramos, A., Silva-Neto, A.J., Llanes-Santiago, O., An approach to fault diagnosis with online detection of novel faults using fuzzy clustering tools, Expert Systems With Applications. 2018; 113:200 - 212.
27. Tuptuk, N., Hailes, S., 2024. Identifying vulnerabilities of industrial control systems using evolutionary multiobjective optimisation. Computer & Security. 2024; 137:1–18.
28. Martí-Coll, A.; Rodríguez-Ramos, A.; Llanes-Santiago, O. An Optimization Approach to Select Koopman Observables for Data-Based Modeling Using Dynamic Mode Decomposition with Control. Processes. 2025; 13:284.
29. Chen, Y., Yin, X., Zhang, R., Gao, F., Reinforced convolutional neural network fault diagnosis of industrial production systems. Chemical Engineering Science. 2024; 299:120466.
30. Zhu, Y., Zhang, R., Gao, F., Industrial process fault diagnosis using dilated convo- lutional stacking bidirectional gated recurrent unit with high and low-order feature fusion. Chemical Engineering Science. 2025; 305:121164.

CONFLICTO DE INTERESES

Ninguno de los autores manifestó la existencia de posibles conflictos de intereses que debieran ser declarados en relación con este artículo.

CONTRIBUCIONES DE LOS AUTORES

Adrián Rodríguez Ramos: Curación de datos, Conceptualización, Investigación, Metodología, Análisis Formal, Software, Validación – Verificación y Redacción.

AUTORES

Adrián Rodríguez Ramos, Ingeniero en Automática. Doctor en Ciencias Técnicas. Profesor Titular de la Universidad Tecnológica de La Habana José Antonio Echeverría (CUJAE). Investigador Visitante del Instituto Politécnico - Universidad del Estado de Rio de Janeiro (UERJ), Email: adrian.ramos@iprj.uerj.br **No ORCID: 0000-0002-0240-7491**. Sus principales intereses de investigación están en el monitoreo de condición y diagnóstico de fallos en procesos industriales usando herramientas de inteligencia computacional.