UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES
INSTITUTO DE POSTGRADO

TÍTULO DEL TRABAJO DE TITULACIÓN

Implementación de un Sistema de Inteligencia Artificial para la Detección, Localización y Alerta de Fallas de Alta Impedancia en Líneas de Distribución Eléctrica de la Subestación Libertad Provincia de Santa Elena

AUTOR

Ronald Ricardo Suárez Reyes

TRABAJO DE TITULACIÓN

Previo a la obtención del grado académico en

MAGISTER EN ELECTRÓNICA Y AUTOMATIZACIÓN

TUTOR

Ing. Edwin Valarezo Añazco, Ph. D.

Santa Elena, Ecuador

Año 2024

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES
INSTITUTO DE POSTGRADO

TRIBUNAL DE SUSTENTACIÓN

Firmado electrónicamente por:
ALICIA GERMANIA ANDRADE VERA

Ing. Alicia Andrade Vera, Mgtr.
COORDINADORA DEL PROGRAMA

Firmado electrónicamente por:
EDWIN GIANNINE VALAREZO ANAZCO

Ing. Edwin Valarezo Añazco, Ph. D.
TUTOR

Firmado electrónicamente por:
LUIS ENRIQUE CHUQUIMARCA JIMENEZ

Ing. Luis Chuquimarca Jiménez, Mgtr.
DOCENTE ESPECIALISTA

Firmado electrónicamente por:
JOSE MIGUEL SANCHEZ AQUINO

Ing. José Sánchez Aquino, Mgtr.
DOCENTE ESPECIALISTA

Firmado electrónicamente por:
MARIA MARGARITA RIVERA GONZALEZ

Abg. María Rivera González, Mgtr.
SECRETARIA
UPSE

UNIVERSIDAD ESTATAL PENÍNSULA
DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES
INSTITUTO DE POSTGRADO

CERTIFICACIÓN

Certifico que luego de haber dirigido científica y técnicamente el desarrollo y estructura final del trabajo, este cumple y se ajusta a los estándares académicos, razón por el cual apruebo en todas sus partes el presente trabajo de titulación que fue realizado en su totalidad por RONALD RICARDO SUÁREZ REYES, como requerimiento para la obtención del título de Magister en Electrónica y Automatización.

TUTOR

Firmado electrónicamente por:
EDWIN GIANNINE
VALAREZO ANAZCO

Ing. Edwin Valarezo Añazco, Ph. D.

Santa Elena, 14 de junio de 2024

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES
INSTITUTO DE POSTGRADO

DECLARACIÓN DE RESPONSABILIDAD

Yo, **RONALD RICARDO SUÁREZ REYES**

DECLARO QUE:

El trabajo de Titulación, Implementación de un sistema de inteligencia artificial para la detección, localización y alerta de fallas de alta impedancia en líneas de distribución eléctrica de la subestación La Libertad Provincia de Santa Elena, previo a la obtención del título en Magister en Electrónica y Automatización, ha sido desarrollado respetando derechos intelectuales de terceros conforme las citas que constan en el documento, cuyas fuentes se incorporan en las referencias o bibliografías. Consecuentemente este trabajo es de mi total autoría.

En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance del Trabajo de Titulación referido.

Santa Elena, 10 de junio de 2024

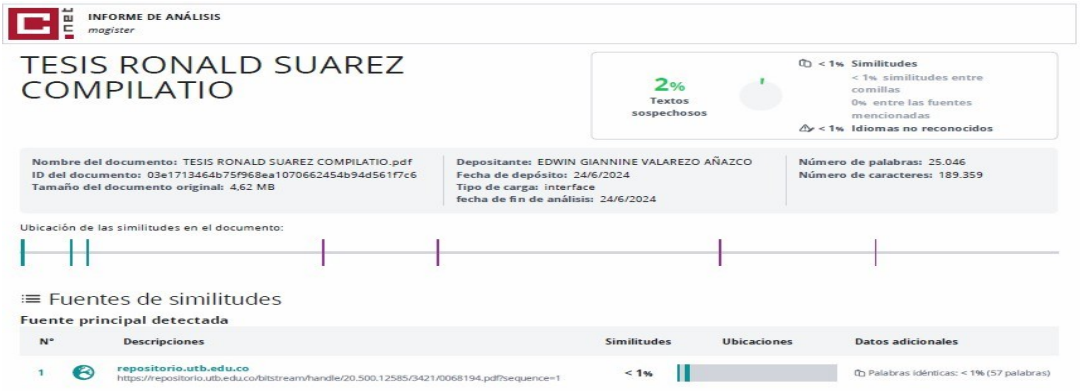
EL AUTOR

Firmado electrónicamente por:
RONALD RICARDO SUAREZ REYES

Ronald Ricardo Suárez Reyes

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES
INSTITUTO DE POSTGRADO
CERTIFICACIÓN DE ANTIPLAGIO

Certifico que después de revisar el documento final del trabajo de titulación denominado Implementación de un sistema de inteligencia artificial para la detección, localización y alerta de fallas de alta impedancia en líneas de distribución eléctrica de la subestación La Libertad Provincia de Santa Elena, presentado por el estudiante, RONALD RICARDO SUÁREZ REYES fue enviado al Sistema Antiplagio COMPILATIO, presentando un porcentaje de similitud correspondiente al 2%, por lo que se aprueba el trabajo para que continúe con el proceso de titulación.


INFORME DE ANÁLISIS
magister

TESIS RONALD SUAREZ COMPILATIO

2% Textos sospechosos

< 1% Similitudes
< 1% similitudes entre comillas
0% entre las fuentes mencionadas
< 1% Idiomas no reconocidos

Nombre del documento: TESIS RONALD SUAREZ COMPILATIO.pdf
ID del documento: 03e1713464b75f968ea1070662454b94d561f7c6
Tamaño del documento original: 4,62 MB

Depositante: EDWIN GIANNINE VALAREZO AÑAZCO
Fecha de depósito: 24/6/2024
Tipo de carga: interface
fecha de fin de análisis: 24/6/2024

Número de palabras: 25.046
Número de caracteres: 189.359

Ubicación de las similitudes en el documento:

Fuentes de similitudes

Fuente principal detectada

N°	Descripciones	Similitudes	Ubicaciones	Datos adicionales
1	repositorio.utb.edu.co https://repositorio.utb.edu.co/bitstream/handle/20.500.12585/3421/0068194.pdf?sequence=1	< 1%		Palabras idénticas: < 1% (57 palabras)

TUTOR

Firmado electrónicamente por:
EDWIN GIANNINE VALAREZO ANAZCO

Ing. Edwin Valarezo Añazco, Ph. D.

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES
INSTITUTO DE POSTGRADO

AUTORIZACIÓN

Yo, **RONALD RICARDO SUÁREZ REYES**

Autorizo a la Universidad Estatal Península de Santa Elena, para que haga de este trabajo de titulación o parte de él, un documento disponible para su lectura consulta y procesos de investigación, según las normas de la Institución.

Cedo los derechos en línea patrimoniales de artículo profesional de alto nivel con fines de difusión pública, además apruebo la reproducción de este artículo académico dentro de las regulaciones de la Universidad, siempre y cuando esta reproducción no suponga una ganancia económica y se realice respetando mis derechos de autor

Santa Elena, 10 de junio de 2024

EL AUTOR

Firmado electrónicamente por:
RONALD RICARDO SUAREZ REYES

Ronald Ricardo Suárez Reyes

AGRADECIMIENTO

A Dios por brindarme fuerzas, salud y sabiduría en cada instante de mi vida, para continuar cosechando logros en la vida.

A la Corporación Nacional de Electricidad Unidad de Negocio Santa Elena, por permitirme desarrollar esta investigación y propuesta, tomando como referencia una de sus subestaciones eléctricas de potencia.

A mi tutor, el ingeniero Edwin Valarezo Añazco por sus valiosos conocimientos, sugerencias y guía de manera experta a lo largo del desarrollo de este proyecto.

Ronald Ricardo, Suárez Reyes

DEDICATORIA

A mis padres, quienes son siempre mi fuente de inspiración en todo momento, pilares fundamentales para alcanzar esta meta tan importante.

A mi hermano Nicolás y hermanas Marianela y la ingeniera Gabriela Suárez por su guía, confianza y apoyo constante durante mis estudios.

Ronald Ricardo, Suárez Reyes

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

En esta propuesta se diseñó e implemento un sistema embebido con inteligencia artificial (IA) para interpretar eventos de alta impedancia y condiciones normales de corriente en un sistema eléctrico de media tensión. La base de datos para el entrenamiento de las redes neuronales (MPL, CNN) y máquina de soporte vectorial (SVM) se generó en MATLAB/Simulink, simulando condiciones reales con un muestreo de 500 muestras por ciclo, de esta forma se usó un 80% para entrenamiento y un 20% para validación. Esto permitió capturar características detalladas para un aprendizaje eficaz. Las pruebas con datos no vistos durante el entrenamiento y reales de eventos de fallas mediante oscilografías obtenidos de relés de protección, mostraron precisiones del 65% al 95%. En el sistema embebido, el tiempo de procesamiento promedio fue de 16.9ms, destacando la viabilidad de la IA para la detección de fallas eléctricas y en entornos críticos donde la precisión es esencial.

Palabras claves: IA, redes neuronales, precisión

ABSTRACT

In this thesis, an embedded system with artificial intelligence (AI) was designed and implemented to identify high impedance events and normal current conditions in a medium voltage electrical system. The database to train the artificial neural networks of multilayer perceptron (MPL), convolutional neural network (CNN), and vector support machine (SVM) were generated in MATLAB/Simulink, simulating real conditions. The sampling frequency was set to 500 samples per cycle. To train, we split the database in 80% and 20% for validation. This allowed to capture detailed features for effective learning. An extra validation with unseen data (i.e., sensed data) and real failure events sensed using an oscillographs obtained from protection relays, showed accuracy from 65% to 95%. In the embedded system, the average processing time was 16.9ms, highlighting the feasibility of AI for electrical failure detection and in critical environments where precision is essential.

Keywords: AI, neural networks, precision

INTRODUCCIÓN

En la industria eléctrica actual, la detección y localización eficiente de fallas en líneas de distribución eléctrica representa un desafío para garantizar la confiabilidad y seguridad del sistema de energía, en este ámbito los sistemas de protección tienen un papel esencial ya que de estas dependen aislar, detectar y localizar eficientemente las fallas que puedan suscitarse en la red eléctrica la cual al estar compuesta por líneas áreas es propensa a sufrir perturbaciones temporales y transitorias (Moloi et al., 2019).

La empresa distribuidora de la energía eléctrica en la provincia de Santa de Elena (CNEL EP UN. STE) cuenta 15 subestaciones eléctricas una de ellas es la subestación Libertad. Esta subestación distribuye el suministro eléctrico a través de 6 alimentadores de energía a diferentes puntos residenciales e industriales del cantón del mismo nombre. Los sistemas de protección eléctrica de estos alimentadores cuentan con las protecciones convencionales de sobre corriente temporizada e instantánea de fase y neutro las mismas que operan a altos niveles de corrientes de cortocircuito y están configuradas mediante los denominados dispositivos electrónicos inteligentes (IEDs). A pesar de ello, la red de distribución es susceptible a ciertas fallas difíciles de detectar, como la falla de alta impedancia, la misma es un tipo de anomalía en la cual los dispositivos de protección convencionales de sobre corriente no detectan debido a que la corriente de este tipo de falla se confunde con la corriente normal de operación del sistema (Vieira et al., 2018). Por lo tanto, resulta crucial desarrollar algoritmos adecuados para la detección e identificación de este tipo de fallas.

Una falla de alta impedancia se describe como la situación en la que hay un contacto eléctrico inesperado entre un conductor energizado y una superficie no conductora de alta impedancia, como una carretera asfaltada, arena, vegetación o ramas de árboles, producto del contacto con estas superficies se produce una corriente muy baja, que oscila desde unos pocos miliamperios (mA) no detectable por la protección de sobre corriente hasta decenas de Amperios (Shihabudheen et al., 2019a). La falta de detección de fallas de alta impedancia conlleva una serie de inconvenientes, como el peligro de descargas eléctricas para las personas cercanas al conductor energizado, así como el riesgo potencial de incendios debido a la formación de arcos eléctricos resultantes de una falla de alta impedancia (Bravo & Pham, 2017). Entre el 5% al 20% de las fallas en el sistema de distribución de la subestación eléctrica en estudio corresponden a fallas de alta impedancia, lo que indica que estas representan un porcentaje significativo. Hoy en día

los fallos producidos por sobre corriente o cortocircuitos han sido despejados con éxitos por los dispositivos electrónicos inteligentes de protección instalados en la subestación, sin embargo ante situaciones en la que un conductor energizado hace contacto con una superficie de alta impedancia estos dispositivos no puede identificarla y por ende no despejarla de manera rápida, ya que estos carecen de algoritmos para detectar fallas de alta impedancia, es por eso que resulta de vital importancia desarrollar algoritmos computacionales que puedan llevar a cabo esta detección con la finalidad de que el sistema de energía siga siendo sostenible, inteligente y confiable.

Planteamiento de la investigación (Fundamentación de la investigación)

La mejora continua en la gestión y operación de la red eléctrica es esencial para garantizar su confiabilidad, sostenibilidad y continuidad. En este contexto, la adopción de tecnologías emergentes en el ámbito eléctrico como la inteligencia artificial (IA) y el análisis de datos son soluciones prometedoras para la detección de fallas que no son posibles detectar con los mecanismos de protección convencionales. Un ejemplo de este tipo de fallas es la falla de alta impedancia, la cual se caracteriza por tener una corriente de falla insuficiente para provocar un disparo en los mecanismos de protección convencionales. No detectar una HIF (High Impedance Fault) en un corto periodo de tiempo puede dar lugar a la ocurrencia de arcos eléctricos sostenidos y graves incendios (Sirojan et al., 2018).

Las fallas de alta impedancia, aunque críticas, presentan un desafío significativo para los métodos de detección de fallas eléctricas convencionales. A pesar de la existencia de dispositivos electrónicos inteligentes de protección instalados en cabecera de los alimentadores de energía eléctrica en las subestaciones y equipos de protección en las líneas de distribución, la identificación de este tipo de fallas sigue siendo un obstáculo debido a que en el mercado existen pocos dispositivos especializados en la detección de este tipo de fallas y a los costos asociados con su adquisición.

El potencial de la inteligencia artificial para analizar patrones complejos en tiempo real ofrece una solución prometedora para esta problemática, dentro de un sistema de aprendizaje supervisado se puede obtener una alta capacidad de identificación de cualquier escenario de falla o cualquier perturbación que no sea de falla, previamente dentro de un conjunto de datos de entrenamiento (Rai, 2021). La IA (Inteligencia artificial) puede detectar de manera eficiente fallas de alta impedancia, permitiendo una

respuesta inmediata que reduce el tiempo de inactividad del servicio eléctrico, disminuyendo la probabilidad de riesgos de incendios, accidentes personales, minimizando así los impactos negativos en los usuarios finales.

Además, la implementación de dispositivos que incluyen tecnologías de IA proporciona una oportunidad única para adoptar un enfoque predictivo en la gestión de la red eléctrica. Al analizar datos históricos y en tiempo real, es posible identificar áreas críticas y anticipar posibles fallas antes de que se conviertan en problemas graves. Esto no solo mejorara la eficiencia de los planes de mantenimiento eléctrico, sino que también allana el camino hacia la evolución de las *Smart Grids* o redes eléctricas inteligentes, representando esto un avance significativo hacia un sistema eléctrico más eficiente, confiable y adaptable a las demandas futuras.

Formulación del problema de investigación

¿Es posible detectar fallas eléctricas de alta impedancia en líneas de distribución de media tensión de la subestación La Libertad, utilizando algoritmos de inteligencia artificial?

Objetivos

Objetivo general

Desarrollar un sistema de detección, localización y alerta temprana de fallas eléctricas de alta impedancia en líneas de distribución de media tensión, mediante el entrenamiento de modelos de inteligencia artificial tales como redes convolucionales y perceptrón multicapa, para mejorar la seguridad, reducir tiempos de respuesta y minimizar las interrupciones en el suministro eléctrico.

Objetivos específicos

- Recolectar el conjunto de datos de fallas de alta impedancia a partir de la simulación en la herramienta MATLAB-Simulink, a través del modelado con parámetros eléctricos reales de un alimentador de distribución eléctrica correspondiente a la subestación La Libertad.
- Desarrollar y entrenar algoritmos de inteligencia artificial que identifiquen patrones de operación nominal, variabilidad de carga y fallas de alta impedancia en líneas de distribución de energía eléctrica.

- Ejecutar pruebas y simulaciones de los modelos de inteligencia artificial para evaluar la precisión y confiabilidad mediante las métricas de evaluación, ante secuencia de datos de operación nominal y fallas simuladas y real extraídas de oscilografías de los equipos de protección instalados en los alimentadores de las subestaciones eléctricas.
- Implementar en un sistema embebido el modelo de inteligencia artificial con las mejores métricas de evaluación para verificar su capacidad de respuesta en tiempo en la detección de fallas de alta impedancia.

Planteamiento hipotético

¿Un sistema de inteligencia artificial permitirá la detección, localización y alerta de fallas de alta impedancia en las líneas de distribución eléctrica de la subestación Libertad en la provincia de Santa Elena?

CAPÍTULO 1. MARCO TEÓRICO REFERENCIAL

1.1 Revisión de literatura

El articulo "*High-impedance fault detection in medium-voltage distribution network using computational intelligence-based classfiers*" de Veerasamy, V., Abdul, N., Ramachandran, R., Thirumeni, M., Subramanian, C., Othman, M., y Hizam, H. (2017) aborda la detección e identificación de fallas de alta impedancia (HIF) en una red de distribución de media tensión de un nivel de voltaje de 13.8kV. Su objetivo principal es evaluar el rendimiento de diferentes clasificadores de inteligencia computacional, como el sistema de inferencia neurodifusa adaptativa (ANFIS) y la máquina de vectores de soporte (SVM), en comparación con otros clasificadores en la detección de fallas de alta impedancia. En la investigación se modela la red del alimentador de distribución eléctrica trifásico en MATLAB/Simulink, obteniendo la señal de corriente de falla del alimentador antes diversas condiciones de falla, incluyendo HIF, falla línea a tierra (L-G), línea a línea (LL), doble línea a tierra (LL-G) y fallas trifásicas, que exhiben una resistencia variable en el tiempo. Posteriormente, dicha señal es sometida a un proceso de muestreo utilizando diferentes niveles de coeficientes de la transformada de wavelet discreta (DWT), en donde se extrae características claves como la desviación estándar (SD), teniendo en cuenta la variación temporal de la impedancia de falla. Dichos valores de SD se emplean en el entrenamiento de los clasificadores difusos, de perceptrón multicapa (MLP), red neuronal bayesiana (BNN), ANFIS y SVM. Finalmente los resultados revelan que los clasificadores ANFIS y SVM son los más efectivos, superando significativamente a los otros clasificadores en términos de rendimiento, logrando tasas de éxito y discriminación del 100%, junto con un bajo error absoluto medio (MAE) y error cuadrático (RMSE), estos resultados sugieren que los clasificadores basados en IA son una opción prometedora para mejorar la confiabilidad y la precisión en la detección de fallas en sistemas eléctricos de distribución (Veerasamy et al., 2019).

En el artículo "*Artificial Intelligence in classifying high impedance faults in Electrical Power Distribution System*", Joga, S., Sinha, P., Maharana, M. (2019), propusieron analizar diferentes clasificadores de redes neuronales probabilísticos (PNN) y de retro propagación (BPNN) para la detección de fallas de alta impedancia. Utilizando la transformada compleja de wavelet de árbol doble o por sus siglas en inglés DT-CWT,

extrajeron características de las formas de onda capturadas en cada bus de un sistema radial de prueba del estándar de bus IEEE33, simulado en MATLAB, capturando las formas de onda en cada bus bajo diversas condiciones operativas, luego aplicaron la técnica DT-CWT para la obtención de los coeficientes relevantes, posteriormente efectuaron la implementación de los clasificadores probabilísticos PNN y BPNN de redes neuronales Los resultados revelaron que el clasificador basado en red neuronal probabilística alcanzó una precisión del 99.41% superando al clasificador basado en retro propagación el cual obtuvo una precisión del 97.65% en la detección de fallas de alta impedancia. Estos hallazgos demuestran la eficacia de un sistema automatizado y preciso para detectar fallas de alta impedancia en el sistema eléctrico (Joga, Sinha & Maharana, 2019).

En su artículo científico titulado "*On the use of artificial intelligence for high impedance fault detection and electrical safety*", Wang, S., y Dehghanian, P. (2020), presentaron en un sistema de monitoreo en línea que emplea aprendizaje automático para detectar de manera rápida y precisa fallas de alta impedancia en sistemas eléctricos. El trabajo propuso una metodología experimental que incluyó la simulación de condiciones de fallas de alta impedancia, la extracción de características utilizando la transformada wavelet de cuadratura pseudocontinua (PSQ-WT), la detección y clasificación de eventos mediante una red neuronal convolucional. Se utilizaron alrededor de 60.000 muestras de formas de onda generadas en el entorno MATLAB/Simulink, con un 80% de los datos para entrenamiento, un 10% para validación y otro 10% para pruebas de la red neuronal convolucional. Esta red fue entrenada durante 120 épocas y logró una precisión de detección general del 99.98%. El tiempo de respuesta del sistema fue de aproximadamente 40ms, compuesto por un retardo de detección de 33ms y un tiempo de procesamiento adicional de 6.3ms para analizar y emitir una respuesta sobre el evento de alta impedancia, representando de esta manera un avance significativo en la seguridad eléctrica (Wang & Dehghanian, 2020).

En el estudio "*High impedance fault detection device based on edge artificial intelligence*" de Bai, H., Lin, Y., Luo, J., Liu, H., Liu, Y. y Li, R. (2023), se desarrolló un dispositivo avanzado de detección de fallas de alta impedancia. Este dispositivo se basa en una CPU Raspberry Pi con una arquitectura ARMv8 de 2GHz y 1GB de memoria RAM. La comunicación entre el componente principal y el módulo de detección de IA se realizó a través de Ethernet de alta velocidad. Durante el proceso de detección, el

dispositivo envía datos de muestreo desde el módulo de adquisición de datos al módulo de algoritmo de IA HIF en tiempo real, en formato de archivos de grabación de ondas, especialmente en formato COMTRADE99. Finalmente, el módulo de IA HIF realiza el cálculo de inteligencia artificial con los archivos de onda grabados y devuelve los resultados del evento al dispositivo HIF en formato de cadena JSON. En las pruebas realizadas en condiciones reales de la red de distribución, el dispositivo de detección de fallas de alta impedancia alcanzó una tasa de identificación correcta del 98.6%, lo que resalta su eficacia en la detección de fallas (Hao et al., 2023).

El estudio "*High Impedance Fault Detection and Localization Using Fully-Connected Convolutional Neural Network: A Deep Learning Approach*" realizado por Abasi-obot, I., Kunya, A., Shehu, G., y Jibril, Y., (2023) propone un enfoque de aprendizaje profundo para la detección y localización de fallas de alta impedancia. En esta investigación, los autores emplearon una red neuronal convolucional completamente conectada para abordar este desafio específico, modelando y simulando eventos de alta impedancia en la red de 33kV/11kV mediante MATLAB. El desarrollo y entrenamiento de la red neuronal convolucional (CNN) se llevaron a cabo utilizando el diseñador de redes profundas disponible en MATLAB, sometiendo la red neuronal a pruebas en diversos escenarios por etapas. Los resultados destacan por su alta precisión, con una tasa de detección del 99.44% y una precisión de localización del 99.78%. Estos resultados representan una mejora significativa, de entre el 2.44% y el 3.96%, en comparación con estudios previos en términos de precisión de detección y localización (Abasi-obot et al., 2023).

1.2 Desarrollo teórico y conceptual

1.2.1 Tipos de fallas en sistemas eléctricos de potencia

Las fallas eléctricas se caracterizan por la desviación anormal del flujo de corriente hacia un camino atípico, lo cual puede causar la interrupción del suministro eléctrico. Estas fallas se clasifican en:

Fallas simétricas (equilibradas) y fallas asimétricas (desequilibradas)

Las fallas simétricas son aquellas que involucran las tres fases. Las fallas trifásicas (L-L-L) y trifásicas a tierra (L-L-L-g) son fallas simétricas.

Las fallas que involucran solo una o dos fases se clasifican como fallas asimétricas o desequilibradas. Las fallas de línea a tierra (L-g), bifásicas (L-L) y bifásicas a tierra (L-L-g) son fallas asimétricas (Oza, Nair, Mehta, & Makwana, 2010).

1.2.2 Sistema de distribución

Un sistema de distribución de energía eléctrica es el conjunto de elementos encargados de conducir la energía desde una subestación de potencia hasta el usuario final o industrial, básicamente, la distribución de energía eléctrica comprende las líneas primarias de distribución, los transformadores de distribución, las líneas secundarias de distribución, las acometidas y medidores de energía los cuales son instalados en la propiedad del cliente residencial o comercial.

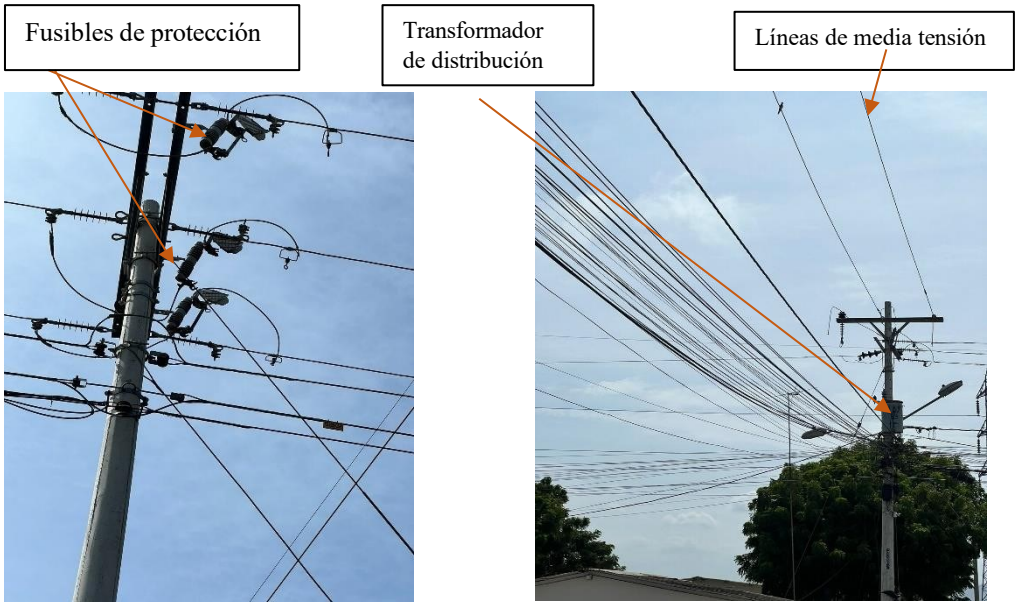


Figura 1. Elementos de un sistema de distribución de energía eléctrica

1.2.3 Objetivos de la distribución de energía eléctrica

La distribución de energía eléctrica debe realizarse de tal manera que el usuario final reciba un servicio sin interrupciones, con un valor de tensión óptimo que le permita el buen funcionamiento de sus equipos eléctricos de manera eficiente, y que la forma de onda senoidal sea pura sin componentes armónicos. La distribución de energía eléctrica debe llevarse a cabo con redes bien diseñadas que soporten el crecimiento de la carga (Yebra, 2010).

1.2.4 Falla de alta impedancia

Las fallas de alta impedancia son un inconveniente en los sistemas de distribución ya que son difíciles de detectar y aislar. En áreas de climas cálidos y terreno seco, los esquemas de protección del sistema de distribución no funcionan adecuadamente cuando las líneas áreas de distribución se rompen y entran en contacto con el piso, esto se debe a que el piso puede estar compuesto de material de alta impedancia que restringe el flujo de la corriente eléctrica a tierra. Por tanto, las corrientes de falla pueden ser muy pequeñas, cercanas a cero. Por definición, una falla de alta impedancia no consume suficiente corriente para hacer que funciones los dispositivos de protección convencionales como disyuntores, relés, fusibles, etc. Estos dispositivos de protección normalmente operan en condiciones de sobrecarga o cortocircuito, de unos pocos cientos a varios miles de amperios respectivamente. Existen en el mercado algunos módulos de relé de alta impedancia disponibles en el mercado eléctrico, pero tienen limitaciones, una de ellas es la mínima corriente de pickup que es de 10 amperios, este valor típicamente es alto para una falla de alta impedancia comúnmente.

1.2.5 Características de una falla de alta impedancia

Las fallas de alta impedancia o HIF por sus siglas en inglés, presentan características complejas que las diferencias de las fallas de cortocircuito normales, dicha complejidad se debe a las siguientes características:

Magnitud de corriente de falla baja: La intensidad de la corriente resulta inferior a las corrientes de pickup de los dispositivos de sobrecorriente, esto debido a la alta impedancia en el punto de falla. La baja magnitud de corriente de falla puede ser difícil de distinguir de un aumento o disminución normal de la carga eléctrica (Milioudis et al., 2012).

Asimetría: La corriente de falla tiene diferentes valores máximos en cada medio ciclo positivo y negativo.

No linealidad: La curva que relaciona las señales sinusoidales de voltaje y corriente durante una falla de alta impedancia es no lineal, esto producto de los diferentes valores de resistividad de las distintas capas de las superficies de contacto.

Buildup y shoulder: Está característica señala que la corriente de falla de una HIF incrementa de manera gradual a lo largo de varios ciclos hasta alcanzar una condición estable (Costa et al., 2015).

La presencia de conductores caídos, usualmente causada por rupturas en los conductores, suele resultar en una disminución de la carga eléctrica o en una sobre corriente temporal, como consecuencia de la caída del conductor que entra en contacto con un objeto conectado a tierra, a menudo se manifiestan con la formación de arcos eléctricos en el punto de contacto.

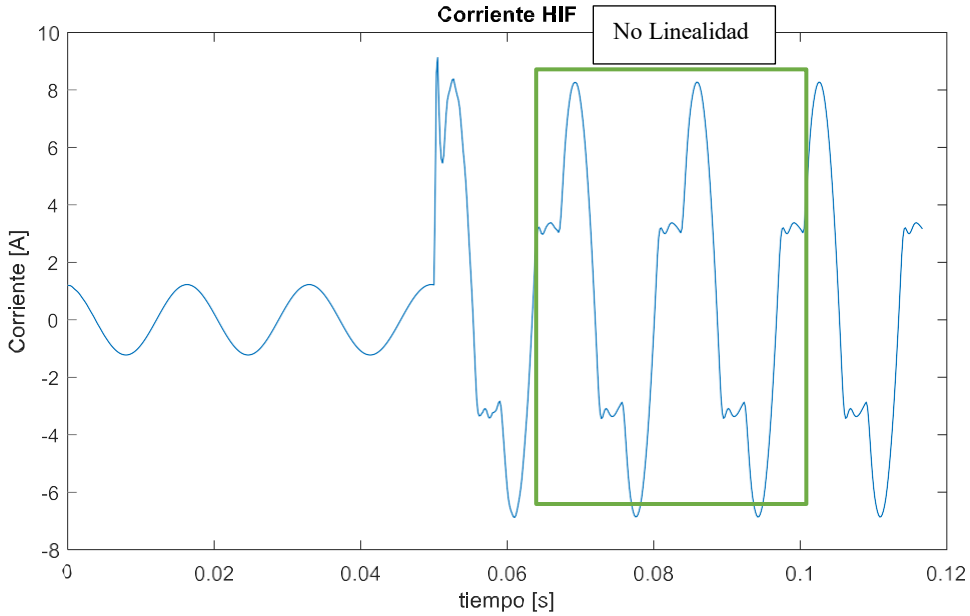


Figura 2. No Linealidad en una falla de alta impedancia

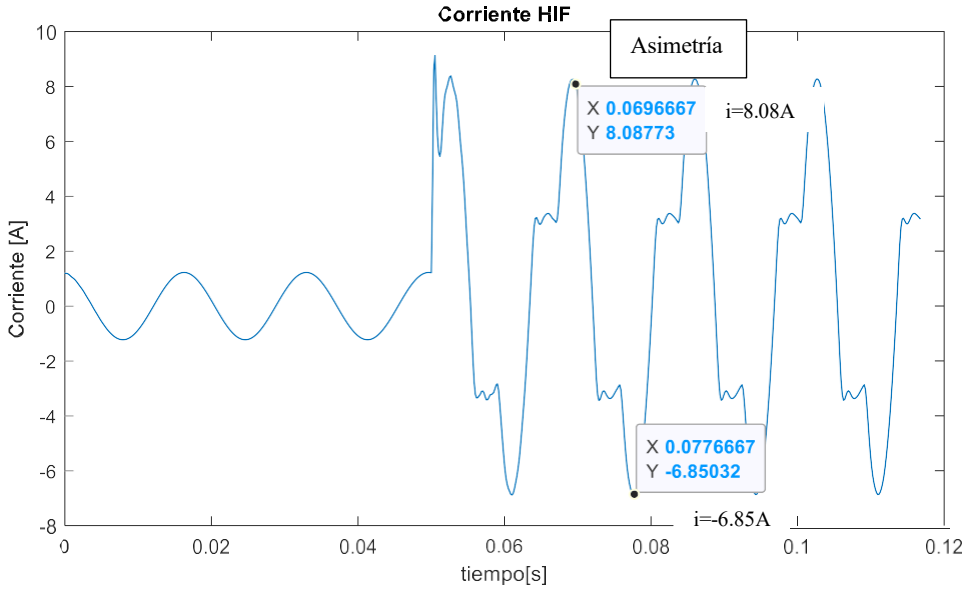


Figura 3. Asimetría en una falla de alta impedancia

1.2.6 Modelos de fallas de alta impedancia

En la investigación se puede encontrar diversas propuestas para modelar las características correspondientes a una falla de alta impedancia, entre los tipos frecuentes se encuentran los modelos eléctricos que emplean elementos como impedancias variables, interruptores controlados y diodos semiconductores, a continuación, se detalla algunos de estos modelos.

Nakamogi en el año 2006 empleó un método sencillo para modelar fallas de alta impedancia (HIF, por sus siglas en inglés). En este enfoque, la falla se describe mediante una resistencia de alto valor óhmico, comúnmente utilizada para representar fallas de baja impedancia. Sin embargo, este modelo carece de la capacidad para reproducir otras características distintivas de las HIF más allá de la baja corriente de falla, lo que limita su utilidad en aplicaciones prácticas.

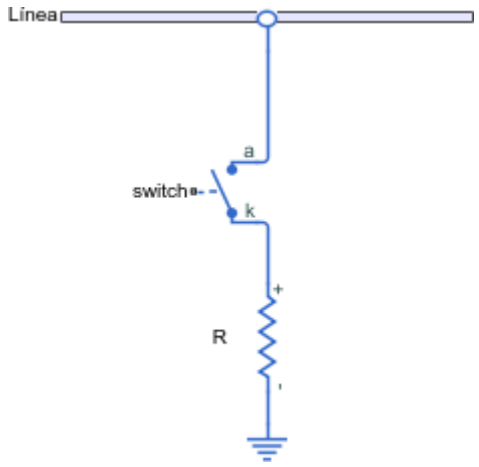


Figura 4. Modelo de HIF basado en Nakamogi (2006)

El modelo de Emanuel (1990) representado en la figura 5, está compuesto por una inductancia y resistencia constante en serie, seguido por la combinación de dos diodos conectados en configuración antiparalelo. Cada diodo está asociado a un generador de voltaje continuo en el cual, durante el semiciclo positivo, la corriente de falla es impulsada por la fuente de tensión V_p, mientras que durante el semiciclo negativo es impulsada por el generador V_n. Dado que la corriente durante el semiciclo positivo es ligeramente mayor que durante el semiciclo negativo, el voltaje V_n será mayor que el voltaje V_p. En este modelo la presencia de contenido armónico está determinada por la diferencia entre las dos fuentes de voltaje ($\Delta V = V_n - V_p$) y la relación entre la reactancia X_L y la resistencia,

se resalta que en este modelo solo se simula el fenómeno de asimetría y no linealidad (Santos et al., 2013).

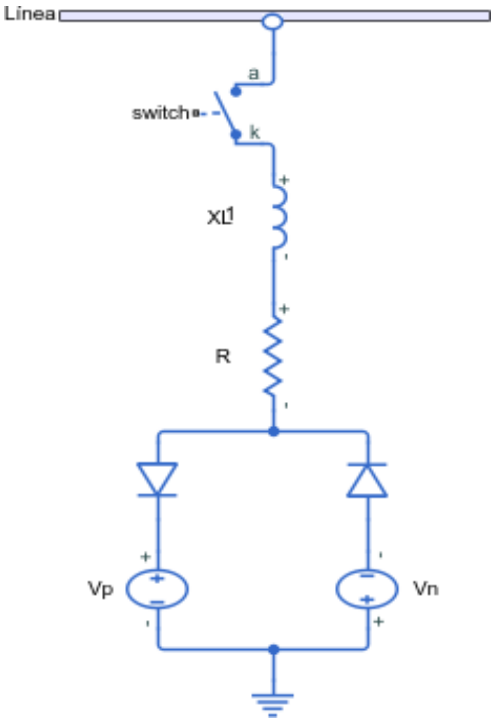


Figura 5. Modelo HIF Emanuel (1990)

En el artículo (Sharaf et al., 1993) para tener en cuenta no linealidades de la impedancia de tierra se conectan dos fuentes de corriente continua, dos resistencias no lineales y dos diodos en conexión antiparalelo como se muestra en la figura

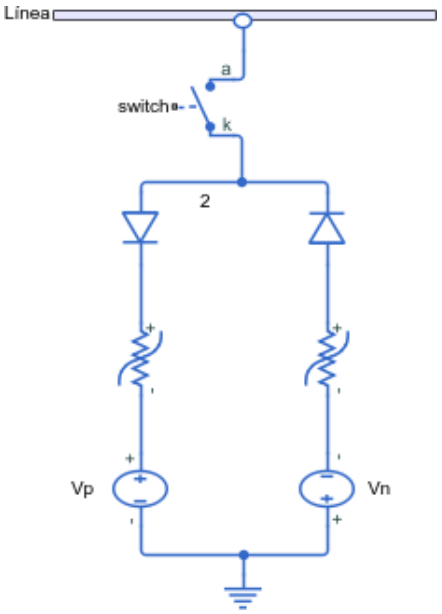


Figura 6. Modelo de HIF mostrado en (Sharaf et al., 1993)

En el trabajo referenciado en (Dery et al., 2017) se introdujo un modelo simplificado de Emanuel, tal como se muestra en la figura 7, el modelo propuesto tiene dos resistencias desiguales que representan la característica de la corriente de falla asimétrica, las dos resistencias Rp y Rp modelan la resistencia de falla.

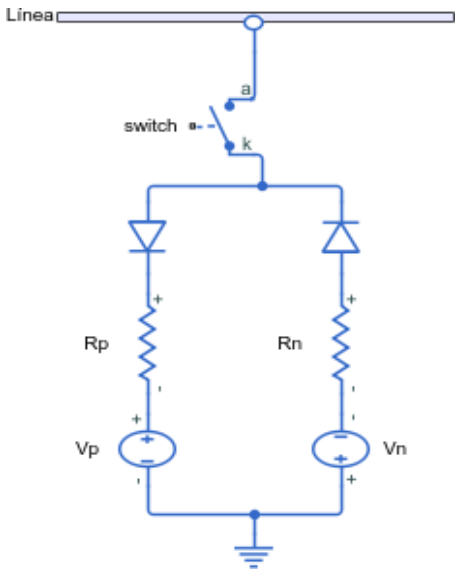


Figura 7. Modelo de HIF mostrado en (Dery et al., 2017)

Otro modelo típico de falla de alta impedancia introduce dos inductancias distintas L_p y L_n, debido a que un porcentaje alto de los eventos de arco eléctrico ocurren en circuitos altamente inductivos, la inclusión de estas inductancias tiene como objetivo representar de manera precisa el ciclo de no linealidad y la forma asimétrica (Córdoba Guzmán, 2018) en la curva Voltaje vs Corriente de una HIF (Zamanan & Sykulski, 2006).

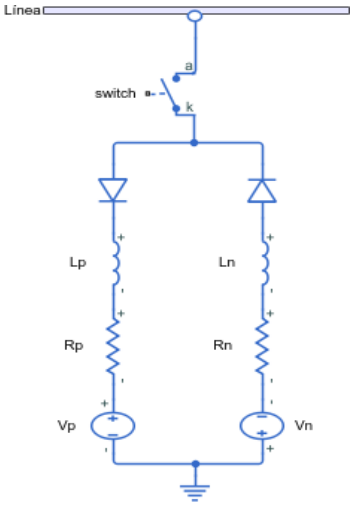


Figura 8. Modelo de HIF con inductancia

1.2.7 Corrientes de falla en varias superficies

Una falla de alta impedancia produce poca o ninguna corriente de falla. Las corrientes de falla varían de 1 a 100 amperios con una forma de onda muy errática. A continuación, se muestran resultados típicos de corriente de falla en dos sitios de prueba para un alimentador de 12.5kV de media tensión (Theron et al., 2018).

Tabla 1. Corriente de falla de alta impedancia en varias superficies

Superficie	**Corriente de falla 7200V L-G**
Asfalto seco	<1 A
Arena seca	<1 A
Asfalto mojado	1A
Arena mojada	15A
Césped seco	20A
Hormigón (no reforzado)	10A
Césped mojado	40A
Hormigón (armado)	75A

Los parámetros del modelo de falla de alta impedancia que representan siete tipos diferentes de superficies de falla se muestran en la siguiente tabla (Ghaderi et al., 2015):

Tabla 2. Parámetros del modelo de falla de alta impedancia para diferentes superficies

Superficie	R1 [Ω]	R2[Ω]	V1[V]	V2[V]
Arena mojada	$138 \pm 10\%$	$138 \pm 10\%$	900 ± 150	750 ± 150
Rama de árbol	$125 \pm 20\%$	$125 \pm 20\%$	1000 ± 100	500 ± 50
Césped seco	$98 \pm 10\%$	$98 \pm 10\%$	1175 ± 175	1000 ± 175
Pasto seco	$70 \pm 10\%$	$70 \pm 10\%$	1400 ± 200	1200 ± 200
Césped mojado	$43 \pm 10\%$	$43 \pm 10\%$	1550 ± 250	1300 ± 250
Hierba mojada	$33 \pm 10\%$	$33 \pm 10\%$	1750 ± 350	1400 ± 350
Hormigón (reforzado)	$23 \pm 10\%$	$23 \pm 10\%$	2000 ± 500	2000 ± 500

1.2.8 Redes neuronales artificiales

Una Red Neuronal se define como una estructura de procesamiento de información que presenta las siguientes características:

- Realiza procesamiento de manera paralela y distribuida

- Está compuesta por elementos de procesamiento conectados entre sí mediante canales de información unidireccionales llamados conexiones
- Cada elemento de procesamiento emite una señal de salida que se distribuye a través de diferentes ramas, según sea necesario
- La señal de salida puede tomar diferente expresión matemática
- Todo el procesamiento de la información se lleva a cabo de manera local y depende exclusivamente de los valores de entrada en el elemento y de los posibles datos almacenados en su memoria local

Una red neuronal presenta las siguientes capacidades:

No linealidad: Indica la capacidad de las redes neuronales de ser no lineales, lo cual es muy importante para modelar sistemas no lineales en problemas complejos.

Mapeo Entradas/Salidas: Proceso de aprendizaje supervisado en el cual la red neuronal aprende los patrones de la entrada y salida deseada, esencial para resolver problemas de clasificación o regresión.

Adaptabilidad: Capacidad de las redes neuronales de ajustar sus pesos sinápticos en respuesta a cambios en el entorno en los datos de entrada, lo que les permite aprender y adaptarse a nuevas situaciones.

Tolerancia a fallos: Rendimiento adaptable de la red neuronal, incluso en presencia de fallos o perturbaciones, lo que las hace robustas y adecuadas para aplicaciones críticas (Moreno et al., s. f.).

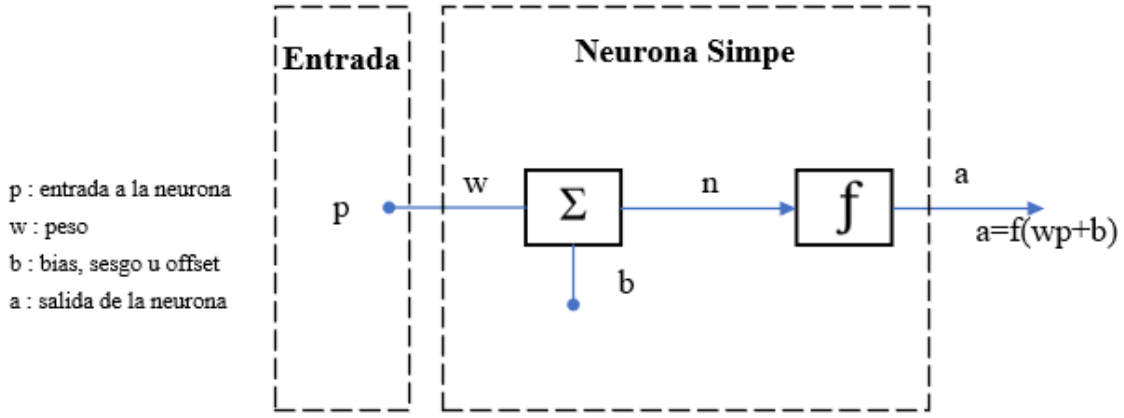


Figura 9. Modelo de una neurona artificial

1.2.9 Estructura de una red neuronal

Una neurona constituye la unidad fundamental de la red y su analogía con una neurona biológica resalta su funcionamiento similar. En la parte superior de la figura 10 se observa una neurona biológica, compuesta por sinapsis, axón, dendritas y cuerpo. Por otro lado, la imagen inferior muestra una neurona artificial, la cual actúa como una unidad de procesamiento de información, realizando cálculos basados en conjunto de vectores de entradas con pesos diferentes (w) para generar una salida única. Con esta información sobre la neurona como la unidad esencial de la red, es posible definir una red neuronal como modelos matemáticos que se inspiran en sistemas biológicos, adaptándolos y simulándolos en entornos computacionales (Asensio & Bowen, s. f.).

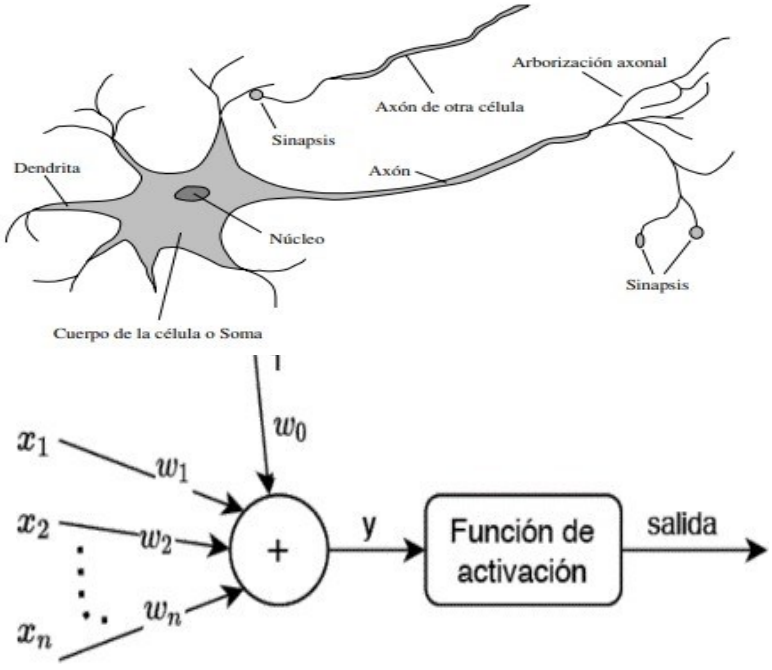


Figura 10. Comparación entre neurona biológica y artificial

1.2.10 Tipos de arquitecturas de redes neuronales

1.2.10.1 Perceptrón Multicapa

La arquitectura Perceptrón Multicapa (MPL de sus siglas en inglés *Multi Layer Perceptrón*), consta de al menos tres niveles de neuronas, la capa de entrada, la capa oculta y la capa de salida. En una red MPL, las conexiones entre las neuronas son

completas. Esto significa que cada neurona de la capa de entrada está conectada con todas las neuronas de la siguiente capa oculta; cada neurona de la capa oculta se conecta con todas las neuronas de las capas anterior y posterior; además, la capa de salida se conecta con todas las neuronas de la capa anterior. Normalmente, las funciones de activación de las neuronas em esta arquitectura son lineales o, con mayor frecuencia, sigmoidales. Una visión clara de esta arquitectura se puede visualizar en la figura 11 (Caicedo B & López S, 2009).

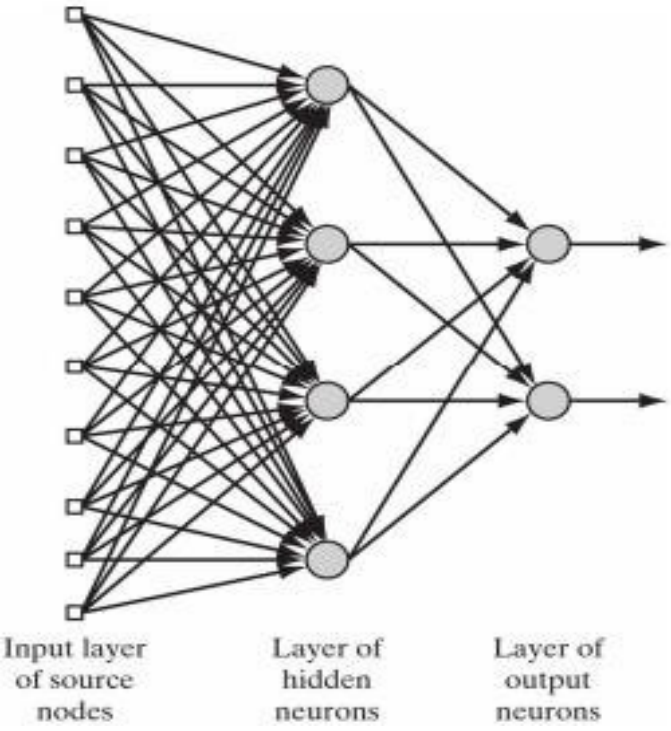


Figura 11. Arquitectura de una red MPL (Haykin, 2009)

1.2.10.2 Redes Neuronales Recurrentes

Las redes neuronales recurrentes (RNN) se caracterizan por la presencia de conexiones retroalimentadas, lo que significa que la información puede circular a través de la red mediante bucles. Estos lazos de retroalimentación pueden establecerse entre neuronas de diferentes capas, entre neuronas de la misma capa o incluso dentro de una misma neurona a lo largo del tiempo. Esta estructura recurrente es especialmente útil para modelar y comprender la dinámica de sistemas no lineales, ya que permite a la red aprender y recordar secuencias temporales o patrones de datos que cambian con el tiempo. En la figura adjunta se muestra un esquema típico de una red neuronal recurrente, donde las

conexiones retroalimentadas están representadas por flechas que indican la propagación de la información a través del tiempo (Serrano et al., 2010).

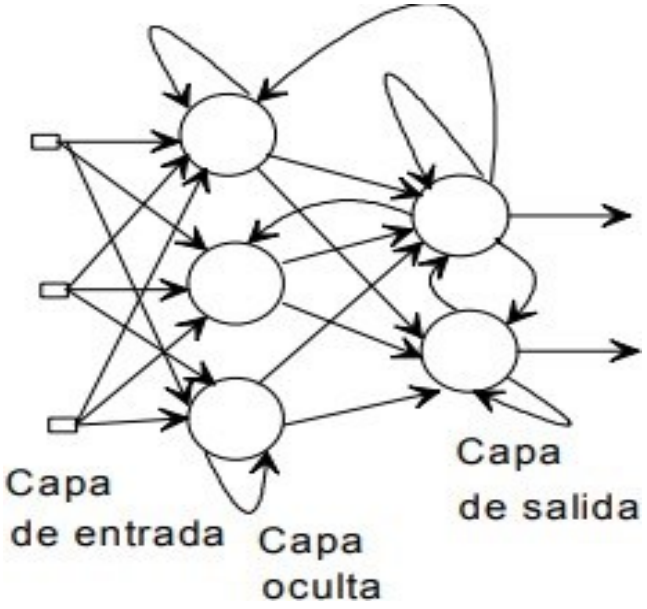


Figura 12. Arquitectura de una RNN

1.2.11 Funciones de activación de redes neuronales

Por mucho que se intente conectar neuronas una detrás de otra, no se lograría una mejora significativa respecto al uso de una sola neurona. Esto se debe a que la ecuación 1 representa una función lineal, y cualquier combinación de funciones lineales, sigue siendo una función lineal.

$$z = \boldsymbol{w}^T\boldsymbol{x} + b = \sum_{i=1}^{N} w_i * x_i + b \qquad \text{Ec.1}$$

Donde x es el vector de entrada a la neurona y el vector w y b son parámetros conocidos como pesos y sesgo respectivamente.

Sin embargo, para introducir un componente que permita obtener un comportamiento global no lineal en la red, es necesario agregar una función adicional al comportamiento de la neurona. Este componente se conoce como función de activación, que son funciones no lineales que toman como entrada la suma ponderada de la neurona z y devuelven la salida final o activación como $a = f(z)$. Existen diversas funciones de activación, siendo las más destacadas la unidad lineal rectificada, la tangente hiperbólica y la sigmoide.

1.2.11.1 Unidad Lineal Rectificada

La función ReLU, también identificada como unidad lineal rectificada, es una función que permanece inalterada para valores de entrada positivos, y devuelve un valor de 0 para entradas negativas. En consecuencia, se caracteriza por ser una función a trozos (Ver ecuación 2), continua y derivable en todo su dominio, a excepción de $x = 0$.

$$f(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases} \quad \text{Ec.2}$$

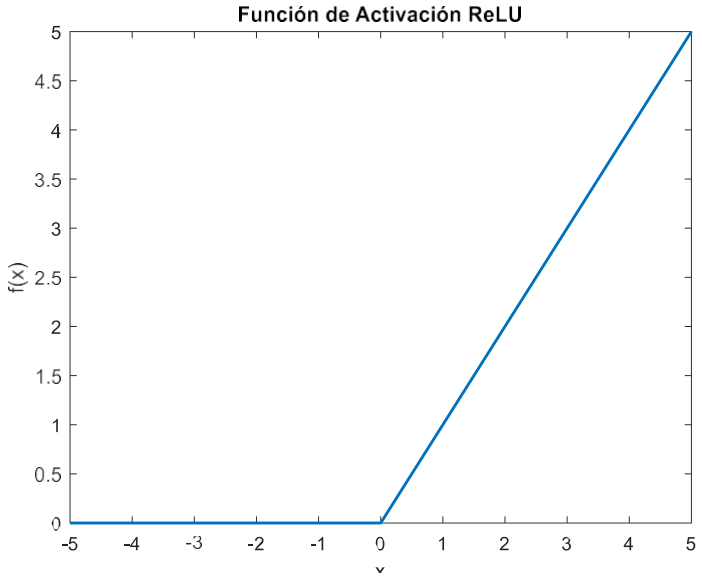


Figura 13. Función de activación ReLU

1.2.11.2 Tangente hiperbólica

La salida de la función hiperbólica está acotada entre -1 y 1, y es simplemente el resultado de aplicar la función trigonométrica tangente hiperbólica.

$$f(x) = \tanh(x) \quad \text{Ec.3}$$

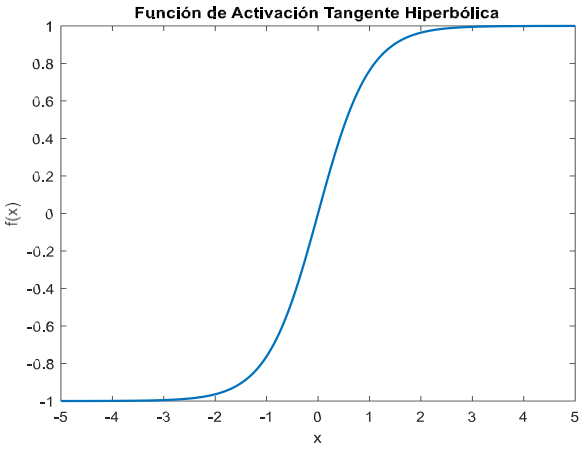


Figura 14. Función de activación tangente hiperbólica

1.2.11.3 Sigmoide

Es habitual encontrar esta función de activación en las capas de salida de las redes neuronales destinadas a clasificación, ya que dicha función limita la salida a valores dentro del rango de 0 a 1 (Pastor Ruiz, 2021), la ecuación 4 muestra la función sigmoide.

$$f(x) = \frac{1}{1+e^{-x}} \qquad \text{Ec.4}$$

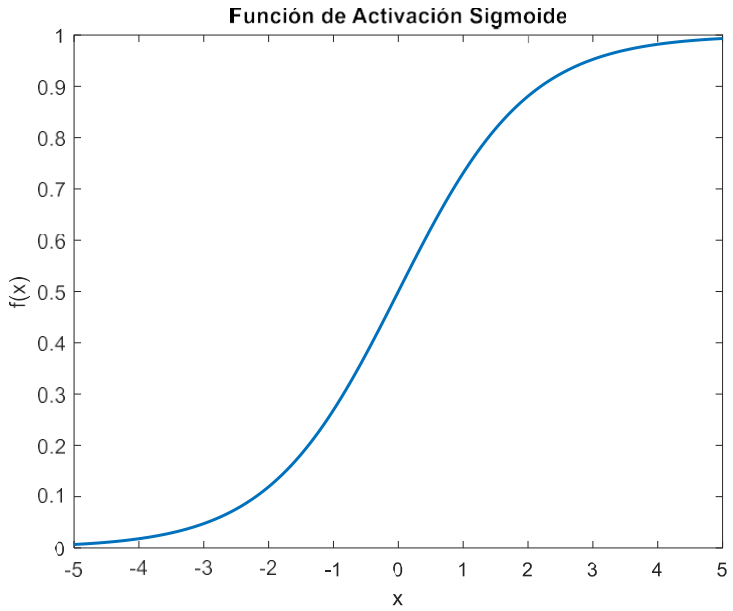


Figura 15. Función de activación sigmoide

1.2.12 Que es la inteligencia artificial

La inteligencia artificial suele definirse como la capacidad de las computadoras para realizar tareas que normalmente requieren inteligencia humana. Sin embargo, una definición más detallada, la IA se puede describir como la habilidad de las máquinas para emplear algoritmos, aprender datos y aplicar ese aprendizaje en la toma de decisiones, emulando así el comportamiento humano. La capacidad de las computadoras y programas para aprender y tomar decisiones es un aspecto de vital importancia de una IA, gracias a estas capacidades, los sistemas de IA pueden realizar una amplia gama de tareas con tasas de errores significativamente menores (Rouhiainen, 2018).

La inteligencia artificial tiene aplicaciones técnicas potenciales que están creciendo en la actualidad:

- Reconocimiento, clasificación y etiquetado de imágenes
- Mantenimiento predictivo

- Procesamiento eficiente y escalable de datos
- Detección y clasificación de objetos

Protección contra amenazas de seguridad cibernética (Rouhiainen, 2018)

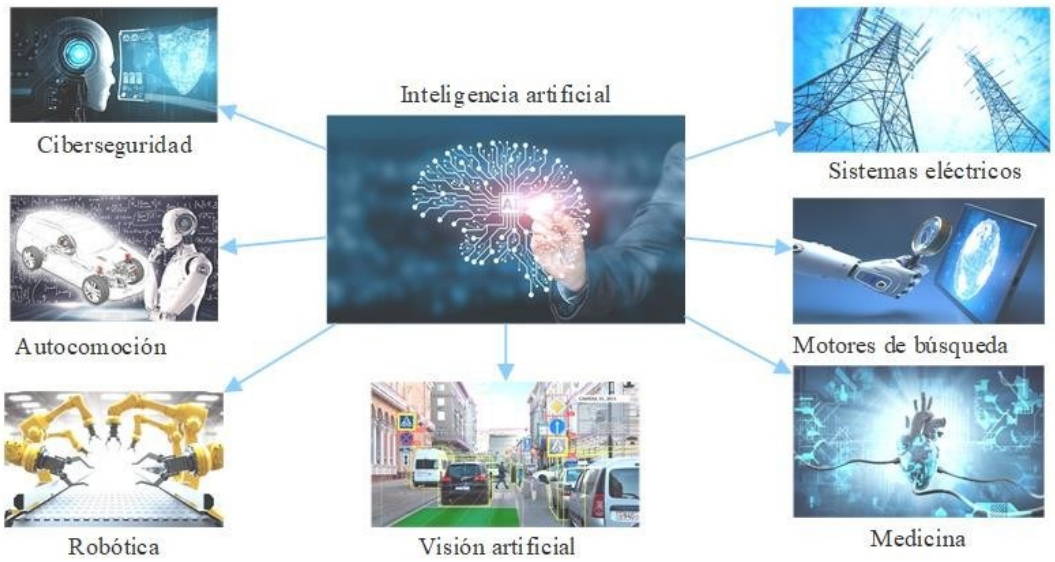


Figura 16. Ámbitos donde se puede utilizar inteligencia artificial

1.2.13 Tipos de Aprendizaje automático

Aprendizaje supervisado: Este enfoque implica entrenamiento con un guía o profesor y utiliza información global. Se proporcionan pares de vectores de entrada y salida deseados. La red calcula una salida que se compara con la salida deseada, y los pesos de la red se ajustan para minimizar el error. Este proceso se repite iterativamente, hasta que la diferencia entre la salida calculada y la deseada sea lo suficientemente pequeña.

Aprendizaje No supervisado: En este caso, no se utiliza un guía o profesor y solo se emplea información local durante el proceso de aprendizaje. Este enfoque se asemeja más al sistema biológico, ya que no se requiere un vector de salida esperado. El algoritmo ajusta los pesos de manera que las salidas sean coherentes, es decir, que, para entradas similares, la red genere salidas similares. Este proceso extrae características al identificar propiedades comunes en el conjunto de datos y agrupa los datos en clases según similitudes.

Aprendizaje de Corrección de Error: En este tipo de aprendizaje, se ajustan los pesos de conexión entre las neuronas artificiales en función de la discrepancia entre los valores deseados y los calculados para cada neurona de la capa de salida.

Aprendizaje por Refuerzo: Similar al aprendizaje de corrección de error, este enfoque también ajusta los pesos de conexión en función del error, pero difiere en cómo se utiliza la información de error. Mientras que el aprendizaje de corrección de error utiliza información de error específica para cada neurona de la capa salida, el aprendizaje por refuerzo utiliza información de error más general para guiar el desarrollo de la red. Este enfoque es adecuado cuando no se dispone de información específica sobre el error, pero se cuenta con información global sobre el rendimiento de la red, como en aplicaciones de predicción y control (Gómez Quesada et al., 1994).

1.2.14 Métricas de análisis

Las métricas en los sistemas de inteligencia artificial permiten evaluar y comparar los algoritmos desarrollados. Estas métricas se fundamentan principalmente en las etiquetas de anomalías y normalidad asignada a cada punto de datos. Sin embargo, aunque los algoritmos entrenados pueden generar una salida booleana, sino una medida numérica de anomalía, indicando cuán diferentes es el punto de dato en comparación con los demás (Lu & Lysecky, 2017).

1.2.14.1 Matriz de confusión

La matriz de confusión es un recurso en sistema de inteligencia artificial, que evalúa como un algoritmo de clasificación se comporta al distribuir los valores reales y las predicciones en cuatros escenarios diferentes, basados en dos variables:

Verdadero o Falso para los valores actuales

Positivo o Negativo para las predicciones

		Predicción	
Observación		Positivo	Negativo
	Positivo	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativo	Falsos Positivos (FP)	Verdaderos Negativos (VN)

Figura 17. Matriz de confusión

Los criterios derivados de la matriz de confusión son los siguientes:

Verdadero Positivo (VP): Ocurre cuando se predice que es positivo y realmente lo es, indicando una clasificación correcta.

Verdadero Negativo (VN): Ocurre cuando se predice que es negativo y realmente lo es, indicando una clasificación correcta.

Falso Positivo (FP): Ocurre cuando se predice que es positivo, pero en realidad es negativo, lo que representa una clasificación incorrecta.

Falso Negativo (FN): Ocurre cuando se predice que es negativo, pero en realidad es positivo, lo que representa una clasificación incorrecta.

1.2.14.2 Sensibilidad

Esta métrica de análisis conocida como "*Recall*" indica la proporción de verdaderos positivos que fueron identificados correctamente por el algoritmo de inteligencia artificial (Powers, 2008). Se determina mediante la ecuación 5:

$$Recall = \frac{VP}{VP+FN} \qquad \text{Ec.5}$$

1.2.14.3 Precisión

También conocida como "*Acurracy*" se refiere al valor que permite conocer el porcentaje de la data de entrada que fue clasificado correctamente. La optima precisión para un modelo viene dada por el valor de 1 mientras que para una pésima precisión es 0 (Choo & Dehghantanha, 2020). Se calcula mediante la siguiente fórmula:

$$Acurracy = \frac{VP+VN}{VN+FP+FN+VP} \qquad \text{Ec.6}$$

1.2.14.4 Especificidad

Representa la proporción de casos negativos que fueron correctamente identificados como negativos. Una alta especificidad indica que el modelo que el modelo tiene menos falsos positivos, es decir, clasifica correctamente la mayoría de los casos negativos (Sokolova et al., 2006). Se calcula mediante la ecuación 7:

$$Especificidad = \frac{VN}{VN+FP} \qquad \text{Ec.7}$$

1.2.14.5 Exactitud

También conocido como "*Precision*" por su término en inglés, evalúa que proporción de las predicciones positivas del modelo son realmente correctas comparándolas con todas las predicciones positivas hechas por el modelo. Esta métrica proporciona una visión de cuán confiables son las predicciones positivas del modelo. Una alta precisión muestra que el modelo está realizando menos predicciones positivas incorrectas, esto indica que hay menos casos en lo que el modelo clasifica incorrectamente un caso negativo como positivo. Mediante la ecuación 8 es posible calcular esta métrica:

$$Precision = \frac{VP}{VP+FP} \quad \text{Ec.8}$$

1.2.14.6 F1-score

La métrica *F1-score* combina precisión y sensibilidad en una sola métrica, lo que la hace muy útil cuando existe una distribución desigual de las clases. Se calcula mediante la siguiente fórmula, como se muestra en la ecuación 9:

$$F1 = 2 \times \frac{Recall \times Presicion}{Recall + Precision} \quad \text{Ec.9}$$

Esta métrica nos proporciona una vision integral de la capacidad de un modelo de inteligencia artificial para manejar diferentes clases, a continuación, muestran los cuatro posibles casos para cada clase:

Alta precisión y sensibilidad: el modelo clasifica correctamente la clase de manera congruente.

Alta precisión y baja sensibilidad: el modelo puede no detectar bien la clase en general, pero cuando lo hace, tiene una alta confiabilidad.

Baja precisión y alta sensibilidad: el modelo puede identificar correctamente la clase, pero al mismo tiempo incluye muestras de otras clases en las predicciones.

Baja precisión y sensibilidad: el modelo no logra clasificar correctamente la clase.

1.2.14.7 ROC-AUC

La curva ROC es un gráfico que muestra cómo un modelo de clasificación se desempeña en diferentes umbrales de decisión. La métrica principal, el AUC (Área Bajo la Curva

ROC), compara modelos al medir cuán bien pueden distinguir entre clases. En la curva ROC, la sensibilidad que indica la tasa de verdaderos positivos se grafica en el eje Y mientras que la especificidad que denota la tasa de verdaderos negativos en el eje X.

La interpretación de esta métrica se divide de la siguiente manera:

Un valor cercano a 1 indica que el modelo es excelente con una gran capacidad para distinguir entre las clases positivas y negativas.

Un valor cercano a 0.5, sugiere que el modelo carece por completo de capacidad para separar las clases, lo que representa el peor escenario, ya que el modelo no puede discriminar correctamente entre la clase positiva y negativa (Steen, 2020).

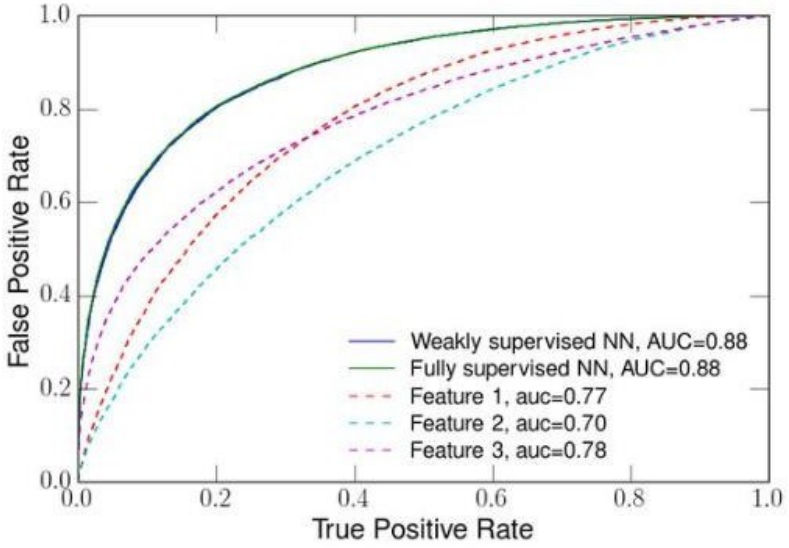


Figura 18. Curva ROC (Dery et al., 2017)

1.2.15 Sistemas embebidos

Un sistema embebido, también conocido como sistema empotrado, es un dispositivo electrónico, diseñado para ejecutar funciones específicas, a diferencias de las computadoras de propósito general que están destinadas a cubrir una variedad más de necesidades. Ejemplos de sistemas embebidos incluyen dispositivos como controles de humedad y sistemas de alarmas. La programación de estos puede realizarse directamente en el lenguaje del microcontrolador, lenguaje C, Python, utilizando compiladores específicos. En ocasiones, se pueden emplear lenguajes orientados a objetos como JAVA, aunque esto no es tan común, ya que los programas de los sistemas embebidos generalmente están diseñados para procesamiento en tiempo real, donde la velocidad de respuesta es crítica (Sánchez & Alberto, 2023).

CAPÍTULO 2. METODOLOGÍA

2.1 Contexto de la investigación

La subestación La Libertad ubicada en la provincia de Santa Elena perteneciente a la cabecera cantonal del cantón que lleva el mismo nombre, es un punto de conexión del sistema eléctrico de la provincia antes menciona, la cual se encarga de transformar el nivel de tensión de alta (69kV) a media tensión (13.8kV) para de esta forma transportar y distribuir la energía mediante las líneas de distribución eléctrica a cada uno de los puntos geográficos de su zona de operatividad (Ver figura 19). Consta de dos transformadores de potencia de 12 y 20MVA de capacidad respectivamente, los cuales permiten suplir la demanda energética a través de los seis alimentadores de distribución y un alimentador de transferencia de carga instalados en cabecera de la subestación eléctrica. Cada uno de los alimentadores de distribución de energía está conformado por un dispositivo electrónico inteligente de medición, protección y el interruptor de potencia (Ver figura 20) el cual interrumpe el paso de corriente ante niveles altos de cortocircuitos, protegiendo de esta manera la instalación eléctrica de cada uno de los alimentadores.

Figura 19. Subestación La Libertad

Figura 20. Tablero de medición, protección, interruptor de potencia alimentadores

Las líneas de distribución energizadas por los respectivos transformadores de potencia de la subestación, representan un eje fundamental en el transporte del suministro eléctrico (Ver figura 21), ya que mediante estas se conectan las cargas de las áreas residenciales, comerciales e industriales a las cuales se debe asegurar la eficiencia, confiabilidad y continuidad del sistema eléctrico, para esto se encuentran instalados dentro de las líneas de distribución reconectadores trifásicos y monofásicos, los cuales no solo protegen contra sobrecorrientes y sobrecargas, sino que también reportan al sistema SCADA las anomalías o fallos en las líneas, teniendo de esta manera una supervisión remota del flujo de energía en campo del sistema eléctrico.

Figura 21. Líneas de distribución eléctrica

2.2 Diseño y alcance de la investigación

La propuesta contempla en primera instancia el modelado del sistema eléctrico de distribución de un alimentador de alta densidad correspondiente a la subestación La Libertad utilizando MATLAB/Simulink con el *toolbox* de elementos *SimPowerSystems*, la cual contiene elementos de potencia importantes tales como fuentes de corriente alterna, transformador de potencia, líneas de transmisión, conductores, sistemas de medición, elementos vitales para el modelado del alimentador de distribución. El modelado considera datos reales de niveles de tensión de la subestación, longitudes y tamaño de los conductores eléctricos, capacidad de los transformadores de distribución, tomados del sistema Geoportal de la Corporación Nacional de Electricidad.

En el diseño de la propuesta se considera un tipo de investigación experimental, considerando la aplicación de un modelo de falla de alta impedancia con diferentes valores de parámetros eléctricos correspondientes a las distintas superficies de contacto a las cuales puede presentarse una falla en diferentes puntos a lo largo del alimentador de distribución, manipulando de forma controlada las diferentes formas de onda sinusoidales de corriente y voltaje, magnitud de corriente RMS nominal y de falla, lo que permitirá tener un conjunto de datos amplio de entrenamiento con el que se pueda desarrollar y evaluar un algoritmo de inteligencia artificial para la detección de estas fallas en particular.

El alcance de la investigación se basará en un enfoque exploratorio y descriptivo. En el primero, se abordará el entrenamiento y desarrollo de las distintas arquitecturas de inteligencia artificial, como las redes neuronales multicapa (MPL), convolucionales (CNN) y máquina de vector de soporte (SVM), estas serán comparadas mediante las diferentes métricas de rendimiento y precisión al evaluar la detección de fallas de alta impedancia. En un segundo enfoque el alcance de la propuesta se basa en describir detalladamente las características esenciales y predominantes en las fallas de alta impedancia, tales como la asimetría de la onda sinusoidal de corriente y la no linealidad obtenida del entorno de simulación MATLAB/Simulink al evaluar el sistema de distribución eléctrica del alimentador en diferentes escenarios de prueba.

2.3 Tipo y métodos de investigación

Esta propuesta mantendrá un enfoque cuantitativo, ya que a través de la simulación del sistema de distribución de uno de los alimentadores de energía en el entorno MATLAB/Simulink, se recolecta las medidas de voltaje y corriente con la finalidad de identificar formas de ondas entre ambas mediciones y patrones de fallas en las ondas sinusoidales que permita de esta manera identificar la presencia de una falla de alta impedancia para posteriormente efectuar una predicción binaria “1 lógico” si existe una falla de alta impedancia presente en la línea de distribución eléctrica o “0 lógico” si se presentó una condición de operación normal del sistema y en otros casos un aumento de demanda eléctrica en el alimentador de energía.

La metodología de la investigación hipotético-deductivo se emplea para determinar si, al utilizar inteligencia artificial específicamente para la detección de fallas de alta impedancia, estas pueden predecirse y detectarse de manera eficiente. Con esto, se busca

probar la hipótesis planteada que permita verificar si las alertas de fallas derivadas de la inteligencia artificial mejorarán la toma de decisiones para el restablecimiento del suministro eléctrico en un tiempo óptimo.

Además, del enfoque hipotético-deductivo se utiliza la metodología analítica ya que se efectúa un análisis de la data obtenida de la simulación del sistema eléctrico de la subestación ante diversos escenarios de variabilidad de carga eléctrica, inyección de fallas de diferentes superficies como césped, ramas de árboles, concreto, para de esta forma extraer las características que permitirán el entrenamiento y desarrollo de un algoritmo de aprendizaje profundo idóneo para el sistema de detección de la falla eléctrica en estudio.

2.4 Población y muestra

La población que se considera en esta propuesta se detalla en la Tabla 3, la cual abarca los alimentadores de distribución eléctrica instalados en la subestación Libertad.

Tabla 3. Población de la propuesta

Transformador	Subestación	Alimentador /Nombre Alimentador	Tipo Alimentador	Extensión Troncal [km]	KVA Instalados
T1 16/20MVA	La Libertad	1/ Acacias	Distribución	3.48km	5MVA
	La Libertad	2/ Propicia	Distribución	3.72km	5MVA
	La Libertad	3/Zona Industrial	Distribución	4.12 km	5MVA
	La Libertad	4/General Enríquez	Distribución	4.36 km	5MVA
T2 10/12.5MVA	La Libertad	5/Libertad	Distribución	3.93km	5MVA
	La Libertad	6/Puerto Nuevo	Transferencia	3.20km	5MVA
	La Libertad	7/Petro Península	Distribución	2.15km	5MVA

La muestra seleccionada para efectuar las simulaciones, adquisición de datos de entrenamiento y características relevantes de las fallas de alta impedancia, consiste en la selección de dos alimentadores de distribución eléctrica, en este caso los siete alimentadores poseen una capacidad instalada de 5MVA por lo que se selecciona dos de alta demanda eléctrica, además se toma en consideración que estos abarcan la zona comercial e industrial del cantón Libertad, zona de vital importancia para el sostenimiento económico de la cabecera cantonal del cantón.

Los alimentadores seleccionados se muestran en la siguiente tabla.

Tabla 4. Muestra seleccionada

Transformador	Subestación	Alimentador /Nombre Alimentador	Tipo Alimentador	Cantidad de Trafos de Distribución
T1 16/20MVA	La Libertad	3/Zona Industrial	Distribución	156
T210/12.5MVA	La Libertad	5/Libertad	Distribución	120

2.5 Estructura de la base de datos para el entrenamiento de la inteligencia artificial

La base de datos que se emplea para el desarrollo del sistema de inteligencia artificial consiste en la recolección de valores muestreados de la onda sinusoidal de corriente de cada una de las fases (A-B-C) del alimentador de distribución eléctrica modelado en MATLAB/Simulink. La frecuencia de muestreo base de la simulación del alimentador será de 50kHz, de esta forma se tendrá señales lo más similares posibles a las continuas.

Teniendo en cuenta esta frecuencia base se adquieren 500 muestras por ciclo de la onda de corriente durante un tiempo aproximado de 16.6 segundos ya que al abarcar este número de muestras es posible captar minuciosamente las características nominales y las características que describen una falla de alta impedancia, obteniendo una simulación de alrededor 1000 ciclos y 500000 muestras en total de cada onda sinusoidal del sistema trifásico. Dentro de esta cantidad de muestras se encuentran incluidas formas de onda de operación normal del flujo de potencia del alimentador y formas de ondas de fallas de alta impedancia, las cuales son separadas mediante ventanas de datos cada dos ciclos. A continuación, en la siguiente tabla se detalla la estructura de la base de datos a utilizar en la propuesta, la cual data de simulación ante diferentes superficies de contacto a la cual se puede producir una falla de alta impedancia.

Tabla 5. Estructura de la base de datos para el entrenamiento del sistema de predicción de fallas

Superficie de falla de alta impedancia	**Ciclos de falla**	**No. de muestras**	**Ciclos de operación nominal**	**No. de muestras**	**Fase**		
					A	**B**	**C**
rama de árbol	480	240000	520	260000	X	X	X
césped seco	480	240000	520	260000			X
pasto seco	480	240000	520	260000		X	
césped mojado	480	240000	520	260000	X		
hierba mojada	480	240000	520	260000		X	
concreto reforzado	480	240000	520	260000	X		
Total, muestras	1440000		1560000				

De la tabla 5 se tiene un total de 1440000 muestras recolectadas que conforman datos con características de fallas de alta impedancia y 1560000 muestras de flujo nominal de carga sin falla en el sistema de distribución, en conjunto esto da un total de 3000000 muestras las cuales serán ingresadas para el desarrollo del sistema de inteligencia artificial, teniendo en consideración que 2400000 son para el entrenamiento del sistema y 600000 para el testeo, de esta forma se tiene un 80% para *train* y un 20% para *test* del sistema de inteligencia artificial.

Tabla 6. Ventanas de falla y operación nominal para el entrenamiento del sistema de predicción de fallas

Superficie de falla de alta impedancia	**Falla de alta impedancia**	**Flujo de carga nominal**
	No. de ventanas de datos	**No. de ventanas de datos**
rama de árbol	480	520
césped seco	480	520
pasto seco	480	520
césped mojado	480	520
hierba mojada	480	520
concreto reforzado	480	520
Total, ventanas de datos	2880	3120

2.6 Metodología de desarrollo

La metodología que se lleva a cabo en esta propuesta consta de cuatros fases las cuales se detallan a continuación:

2.6.1 Fase 1: Simulación y visualización de datos del sistema eléctrico en MATLAB/Simulink

Previo al desarrollo de la primera fase se efectúa la búsqueda de información en lo que concierne a los parámetros eléctricos de la subestación Libertad, tales como numero de alimentadores de la subestación, número y capacidad de transformadores de potencias, calibre del conductor, impedancias de la línea de distribución, extensión en kilómetros de los alimentadores, zonas de operatividad de la subestación y carga instalada a lo largo del alimentador de energía. Teniendo esta información de vital importancia se desarrolla en el software MATLAB/Simulink el modelado del sistema de distribución de energía de los alimentadores mediante el uso de bloques del *Toolboox SimPowerSystems* el cual permitirá simular sistemas de potencia y de esta forma simular flujos de carga, en diferentes escenarios como operación nominal de carga en un 50%, 75%, 100% y condiciones de falla de alta impedancia considerando varias superficies a la que el conductor energizado puede tener contacto, con esto se procura abarcar una gran cantidad de escenarios en condicionales nominales y de falla, formas de ondas sinusoidales de las variables eléctricas de voltaje, corriente y conjuntos de datos que permitan el entrenamiento y desarrollo del sistema de predicción de fallas de alta impedancia.

2.6.2 Fase 2: Procesamiento de datos

En esta fase denominada Procesamiento de Datos, los datos obtenidos de las simulaciones efectuadas en MATLAB/Simulink, se almacenan en una estructura de datos con extensión .mat. Esta estructura contendrá las muestras correspondientes a las variables de corrientes de las fases A-B-C, así como las variables de salida codificadas de manera binaria, donde “0” indica ausencia de falla y “1” presencia de falla.

Una vez obtenida esta estructura de datos, se procede al procesamiento mediante codificación en MATLAB para separar cada una de las variables eléctricas mediante ventanas de datos cada dos ciclos, con un solapamiento entre ellas cada ciclo (Ver figura 22). Este enfoque tiene como objetivo capturar características relevantes de la onda

sinusoidal tanto en condiciones de operación nominal como de falla. Además, cada ventana de datos contiene una salida, lo que facilita la detección de anomalías y condiciones de fallas durante el análisis entre una ventana actual y la siguiente.

La figura 23 muestra de manera global la descripción del conjunto de datos que se utiliza para el entrenamiento de la IA, donde cada ventana de datos es procesada y guardada de forma matricial y se exporta a un archivo .csv. Este archivo contendrá múltiples tensores con los datos de entrada de la variable eléctrica de corriente y su correspondiente salida binaria, estos datos son utilizados para el entrenamiento del sistema de inteligencia artificial.

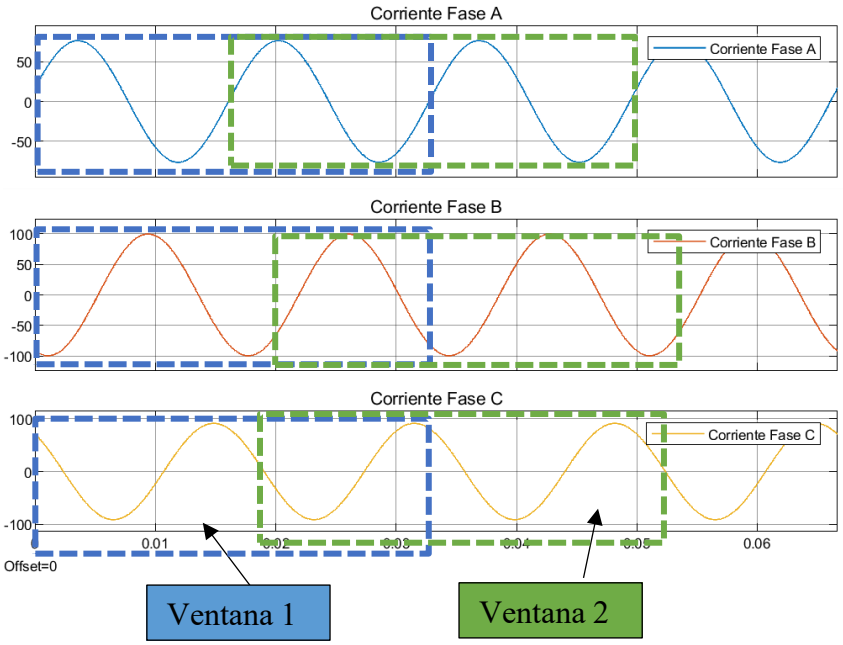


Figura 22. Separado de datos mediante ventanas (overlap 50%)

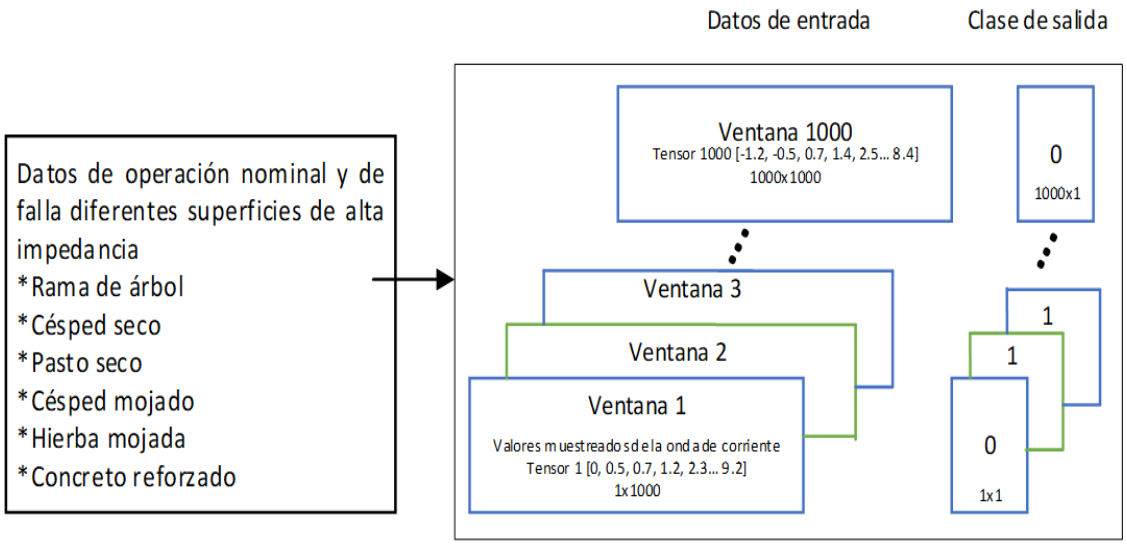


Figura 23. Descripción dataset para entrenamiento del sistema de predicción

2.6.3 Fase 3: Desarrollo, entrenamiento y evaluación del sistema de inteligencia artificial

La tercera fase de la propuesta comprende el desarrollo, entrenamiento y evaluación del modelo de predicción de fallas de alta impedancia. Esta etapa se lleva a cabo mediante el uso de la herramienta Google Colab para el desarrollo y ejecución de codificación Python, en este lenguaje se usa las librerías de *keras, tensorflow*, para diseñar arquitecturas de redes neuronales específicas para esta aplicación eléctrica tales como MPL, CNN y SVM.

Inicialmente se desarrollará una red MPL, seguida de una CNN y SVM , donde se ajustará la arquitectura variando las capas ocultas, intermedias y funciones de activación. Una vez definidas las arquitecturas el 80% de los datos de entrada serán utilizados para entrenar los modelos en un intervalo entre 100 a 150 épocas con la finalidad que el modelo aprenda patrones entre los valores de entrada y la salida utilizando el optimizador "*Adam*", que ajusta automáticamente las tasas de aprendizaje.

Al finalizar el entrenamiento se evaluará el rendimiento de los modelos de inteligencia artificial obtenidos, utilizando métricas como precisión, perdidas, matrices de confusión, curvas de aprendizaje. Además, se incorporarán nuevos datos de operación normal y de falla, para determinar el modelo óptimo capaz de detectar y clasificar las fallas eléctricas en diferentes escenarios. Es importante mencionar que esta fase es un proceso iterativo el cual garantiza la obtención de un modelo confiable y con alta precisión para la detección de las fallas de alta impedancia en sistemas de potencia.

2.6.4 Fase 4: Despliegue del modelo de inteligencia artificial

Esta última etapa del proyecto se centra en el despliegue del modelo de inteligencia artificial en un sistema embebido tal como una tarjeta Raspberry Pi 4, en esta mediante codificación Python se cargará el modelo de inteligencia artificial óptimo (archivo .pb) para leer secuencias de datos de oscilografías recolectadas de relés de protección de fallas reales de alta impedancia suscitadas en las líneas de distribución durante el último trimestre.

Durante estas pruebas se medirá el tiempo de procesamiento de datos y tiempo necesario para que el sistema realice predicciones de la clase de salida. Además, se efectuarán pruebas con un relé de protección, aplicando corrientes nominales y de fallas de cortocircuito mediante una maleta de inyección de corrientes secundaria. Esto permitirá

verificar la capacidad del modelo para operar en condiciones normales y fallas de diferentes características a las que el modelo no fue entrenado originalmente.

Estas pruebas son cruciales para validar el rendimiento del modelo de inteligencia artificial ante situaciones críticas y reales de operación, lo que permitirá asegurar su eficiencia en la detección y clasificación de las fallas eléctricas de alta impedancia en tiempo real.

La figura 24 y 25 muestran el diagrama de flujo y de manera detallada las fases a desarrollar respectivamente en la presente propuesta.

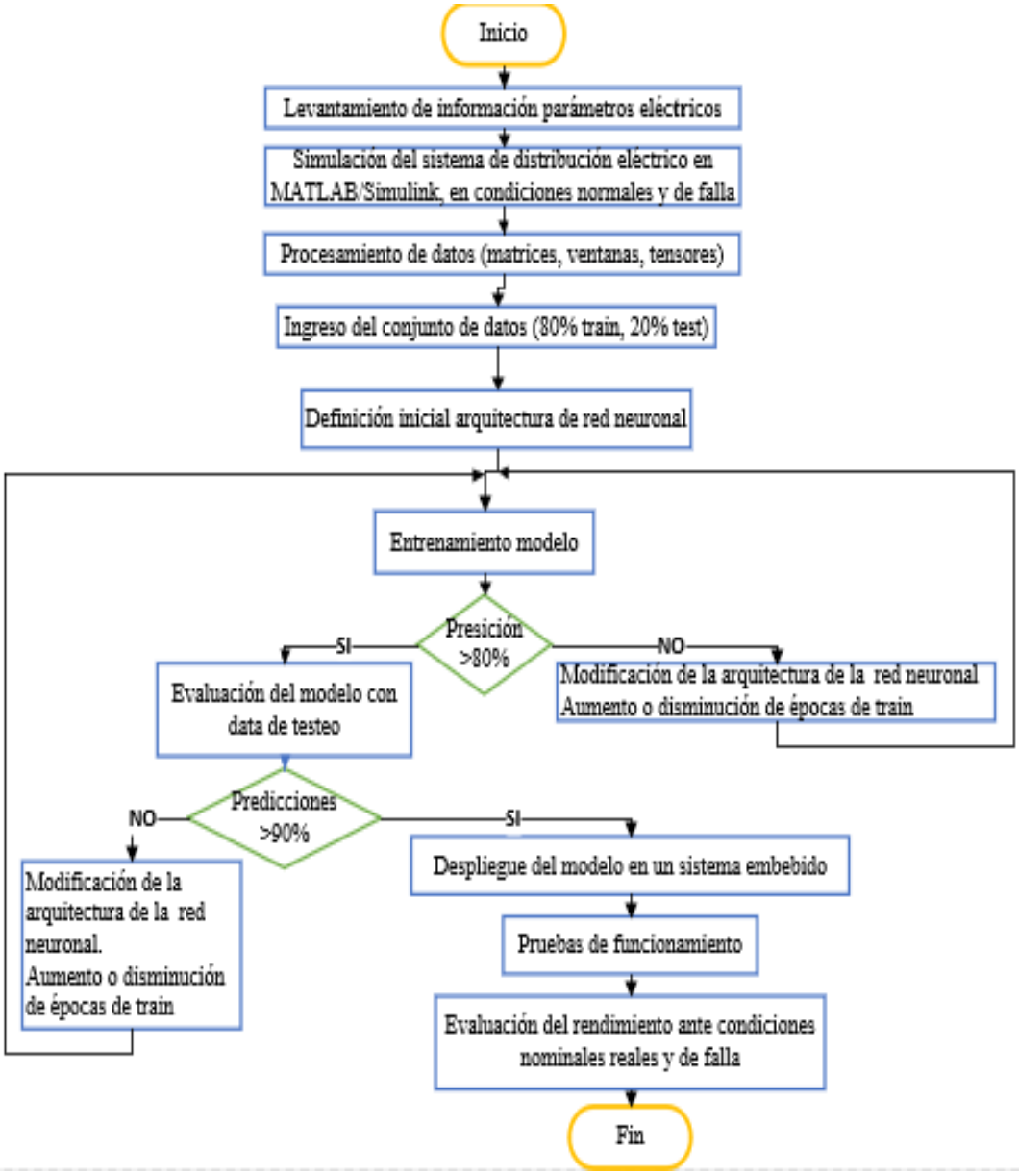


Figura 24. Diagrama de flujo fases de la propuesta

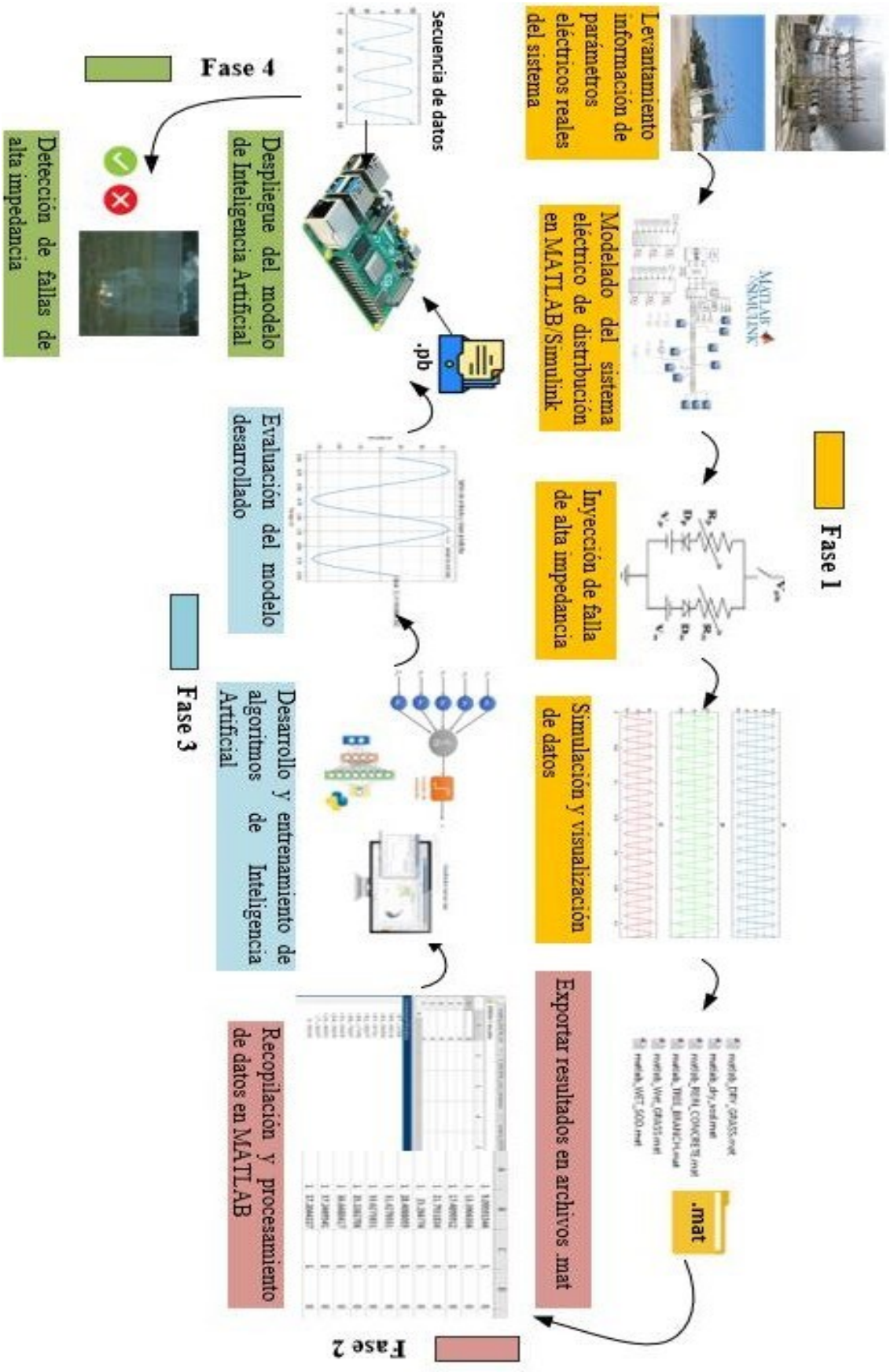


Figura 25. Fases de la propuesta

CAPÍTULO 3. DESARROLLO DE LA PROPUESTA

3.1 Análisis del sistema eléctrico de distribución de la subestación La Libertad

Para desarrollar el sistema de inteligencia artificial capaz de detectar fallas de alta impedancia en las líneas de distribución eléctrica, es fundamental evaluar inicialmente la demanda promedio de corriente eléctrica que presentan los alimentadores de energía eléctrica de la subestación. En la figura 26, se visualizan las corrientes promedio durante el mes de enero de 2024 de todos los alimentadores, en donde la demanda promedio de amperaje abarca entre los 60A y 100 respectivamente. Se opta por seleccionar el alimentador Zona Industrial el cuál bordea una demanda de corriente entre los 77A y 100. Además, dicho alimentador suministra energía al sector industrial y residencial del cantón la Libertad, por lo que es seleccionado para desarrollarlo en el entorno de simulación MATLAB/Simulink, y posteriormente simular fallas de alta impedancia en diferentes localizaciones del alimentador y condiciones de demanda eléctrica.

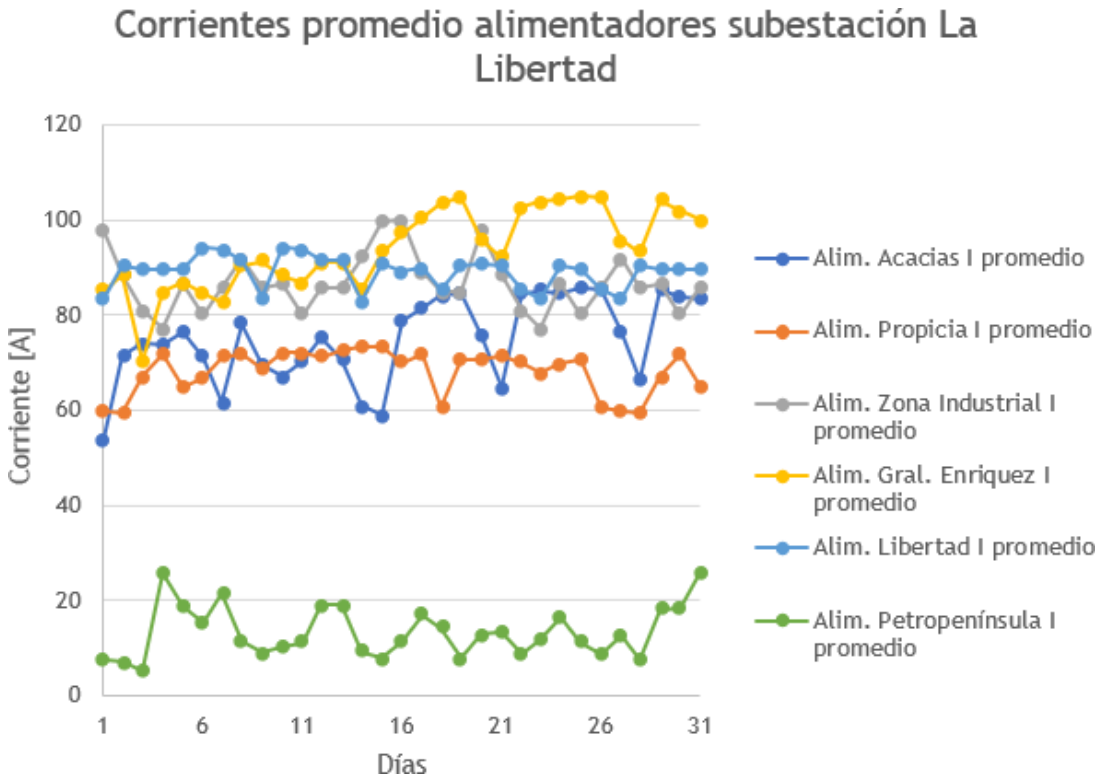


Figura 26. Demanda de corriente alimentadores correspondiente al mes de enero 2024

En el modelado del alimentador Zona Industrial se consideran datos reales tomados del sistema geoportal, tales como capacidad y fase de conexión de los transformadores de potencia de distribución, distancia en kilómetros del conductor de la troncal principal y el respectivo calibre del conductor.

El alimentador Zona Industrial suministra energía eléctrica a los clientes residenciales e industriales del cantón La Libertad, es un alimentador urbano con una extensión de la troncal de aproximadamente 4.12 km, posee una potencia instalada de 4MVA, mientras que la demanda máxima considerada en enero de 2024 fue de 2876 kVA respectivamente. Las características principales de este alimentador se muestran la tabla 7.

Tabla 7. Características del alimentador Zona Industrial

Características principales del alimentador Zona Industrial	
Característica	**Descripción**
Alimentador	Zona Industrial
Subestación	La Libertad
Extensión	4.12 Km
Conductor de la troncal	4/0 AWG
MVA instalado	4 MVA
Factor de potencia	0.97
Demanda promedio	2000 kVA
Demanda máxima	2876 kVA

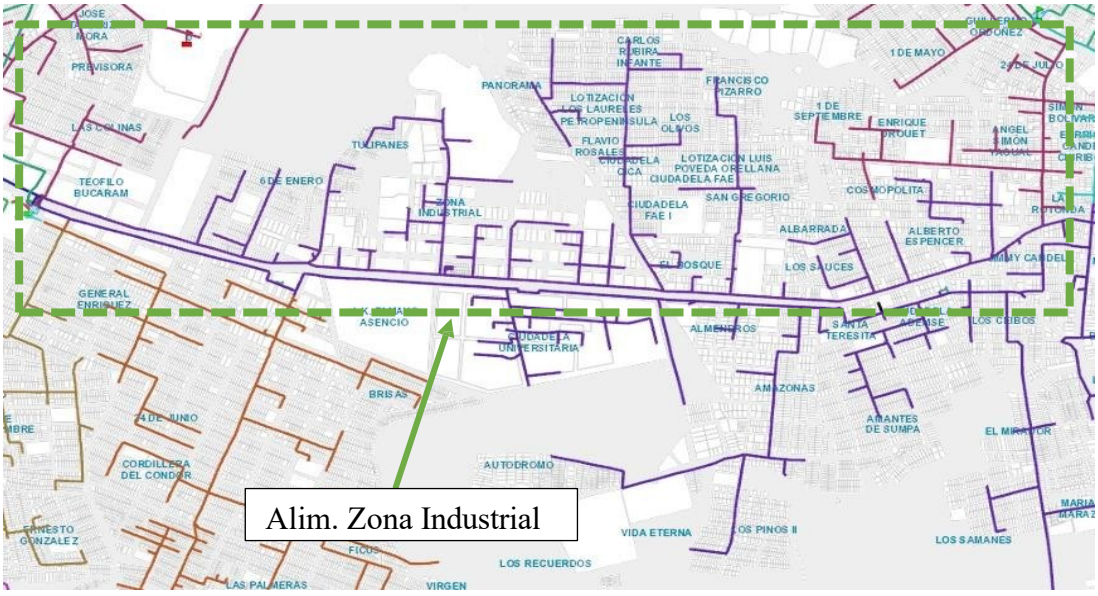


Figura 27.Distribución geográfica de la zona de cobertura del alimentador Zona Industrial

3.2 Modelo para simular fallas de alta impedancia

El estudio de las fallas de alta impedancia no es una tarea fácil ya que recolectar data real de este tipo de fallas incluye acceder a data histórica de eventos de tal magnitud suscitados anteriormente, o por consiguiente ensayar con esquemas reales de protección eléctrica y conductores energizados suele ser una tarea con un grado de dificultad. Por lo tanto, el uso de modelos que permitan obtener las características relevantes que se presentan en una forma de onda de una falla alta impedancia, emerge como una solución viable para representar las características de este fenómeno, las cuales deben estar bien representadas (Ramiro, s. f.), además varias investigaciones relevantes tienen como base el modelo basado en diodos (Shihabudheen et al., 2019b), (Kujur & Biswal, 2017).

En esta propuesta se opta por el modelo de Emanuel (Sekar & Mohanty, 2017) el cual consiste en dos diodos, dos fuentes de tensión de corriente continua y dos resistencias no lineales, ya que dicho modelo representa adecuadamente las características de no linealidad y asimetría, características presentes en un evento de alta impedancia. En esta configuración mostrada en la figura 28, la conexión antiparalela con dos diodos en serie, permiten simular los periodos cero que ocurren durante la asimetría y el arco eléctrico, mientras que las dos resistencias y las fuentes de tensión de corriente continua de diferentes valores permiten representar la naturaleza de la asimetría de la corriente de una falla de alta impedancia. Además, se utilizan los parámetros de la tabla 2, en donde se simulará las diferentes superficies no conductoras a las cuales puede estar expuesto el conductor energizado, para de esta forma tener un conjunto general que representen estas fallas eléctricas.

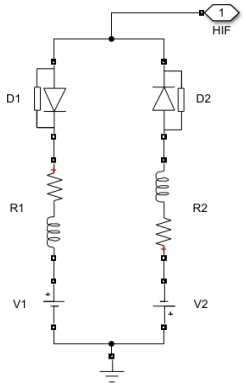


Figura 28. Modelo seleccionado para simular fallas de alta impedancia

El esquema para obtener los datos que contienen las características de una falla de alta impedancia se muestra en la figura 29, la misma que ilustra la forma en que se inyecta la

falla en un punto de la línea de la troncal del alimentador, en diferentes ubicaciones, las mismas que se han considero al 25%, 50% o 75% de extensión de la troncal principal del alimentador, donde se resalta que todas las perturbaciones a simular son fallas de alta impedancia monofásicas a tierra.

Figura 29. Esquema para simular una falla de alta impedancia en la línea de distribución del alimentador

3.3 Sistema eléctrico del alimentador seleccionado en la herramienta MATLAB/Simulink

El modelado del alimentador eléctrico de distribución Zona Industrial, se efectúa en la herramienta MATLAB/Simulink, en este entorno se procede a construir el sistema eléctrico mediante los respectivos bloques disponibles en la librería *SimPowerSystem*, a continuación se enumera los respectivos bloques a utilizar y su posterior configuración en el entorno de simulación, para de esta manera obtener una data sintética lo más cercano a lo real, que contenga datos de una normal corriente de operación y de una onda de corriente que se produce en un fenómeno de falla de alta impedancia:

- *Three Phase Source*
- *Three Phase Transformer (Two windings)*
- *Voltage y current measurement*
- *Three Phase Pi Section Line*
- *Series RLC Load*

- *Linear Transformer*
- *Scope*

En el sistema eléctrico de potencia de la provincia de Santa Elena, a nivel de las líneas de subtransmisión, se maneja un alto voltaje de 69000V (69kV) línea a línea, a una frecuencia de 60Hz. Mediante el bloque Three-Phase Source mostrado la siguiente figura 30, es posible modelar esta fuente de alimentación trifásica con dicho nivel de tensión que ingresa a la subestación eléctrica.

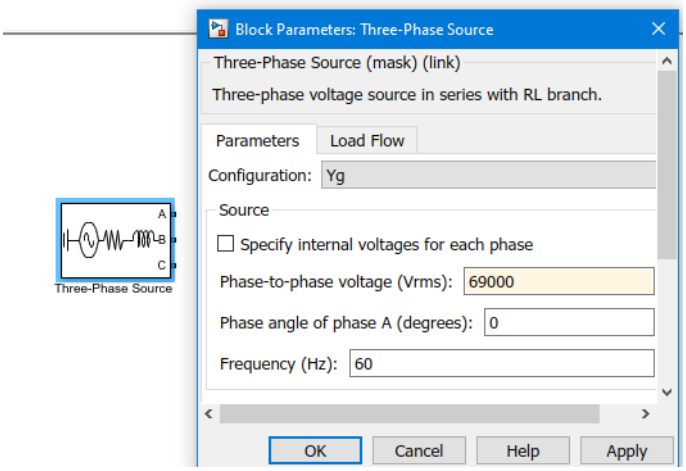


Figura 30. Configuración del bloque de suministro eléctrico en alta tensión

Este voltaje de alta tensión es reducido a un nivel más bajo, en este caso un nivel de media tensión de 13.800V (13.8kV), mediante el transformador de potencia instalado en la subestación eléctrica (Ver figura 32), para de esta forma distribuir la energía a través de las redes de distribución eléctrica. Este transformador es trifásico de una potencia nominal 20000000 VA (20MVA), con una tensión nominal primaria 69kV y una tensión nominal secundaria 13.8kV, operando a una frecuencia 60Hz. El transformador es de tipo exterior y tiene una conexión Dyn1.

La conexión Dyn1 indica que el devanado primario está conectado en delta (D), el secundario en estrella (Y) con el neutro accesible y un desfase de 30 grados entre los voltajes de línea del devanado primario y el secundario (n1).

Es fundamental resaltar que estos parámetros y características del transformador de potencia son importantes para el entorno de simulación mediante el bloque *Three Phase Transformer Two-Windings* (Ver figura 31). La precisión en el modelamiento de los componentes que conforman el sistema eléctrico en estudio asegura una representación

fiel del comportamiento real del sistema, lo que permitiría obtener datos totalmente fiables.

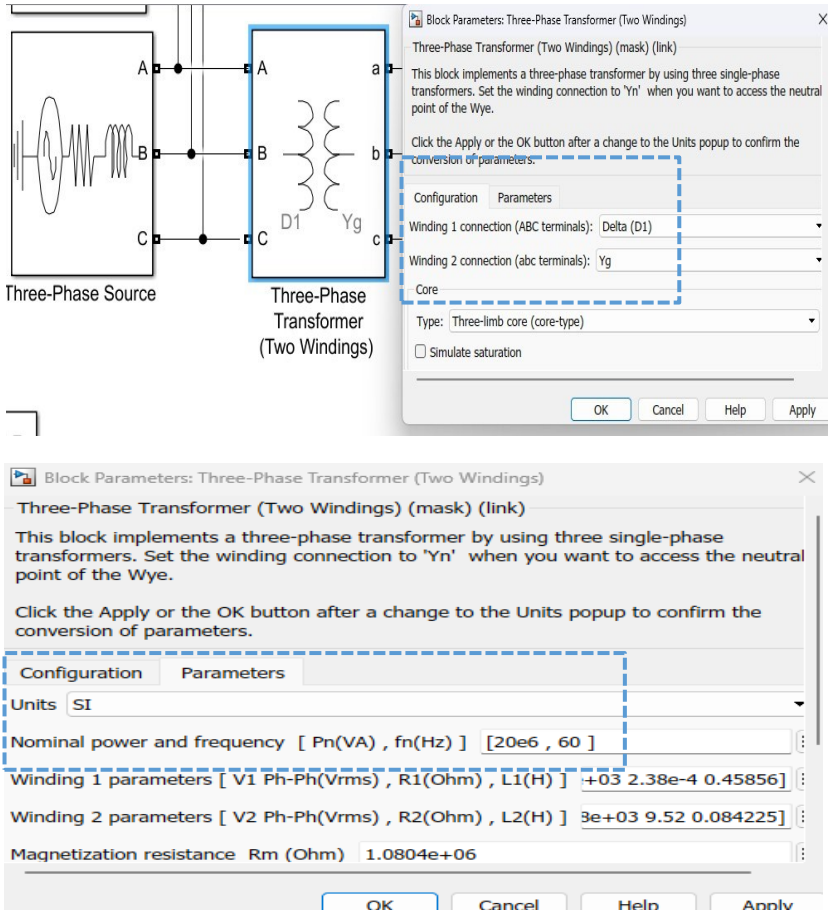


Figura 31. Bloque de simulación, transformador de potencia de dos devanados

Figura 32. Transformador de potencia de la subestación La Libertad

Los respectivos IEDs instalados en las celdas de media tensión (13.8kV) de la subestación son responsables de la medición precisa de corriente y voltaje de cada alimentador, utilizando los datos proporcionados por los transformadores de corriente (TC) y transformadores de potencial (TP) correspondientes (Ver figura 34). Estos IEDs procesan las señales analógicas de los TC y TP, convirtiéndolas en datos digitales que pueden ser analizados y monitoreados. Parra llevar estos datos al entorno de simulación y obtener los valores muestreados, se emplean bloques de medición de variables eléctricas como *current* y voltaje *measurement* (Ver figura 33). Estos bloques de medición son cruciales en el entorno de simulación, ya que con estos se adquiere toda la respectiva data que se ingresara en el entrenamiento de las redes neuronales.

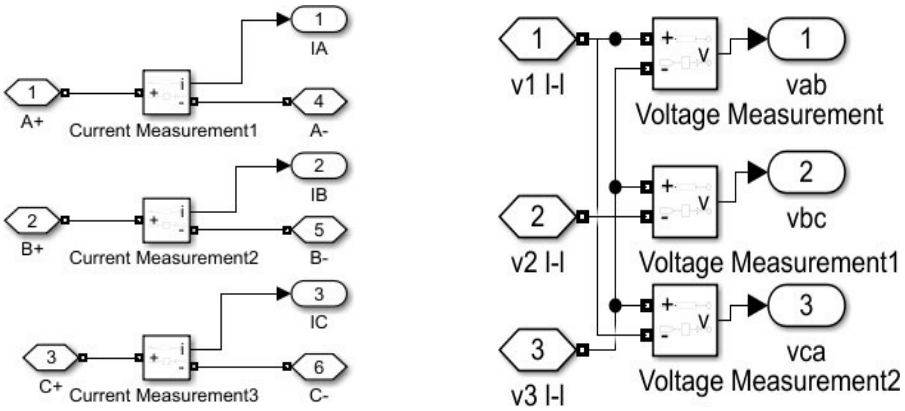


Figura 33. Sistemas de medición de amperaje y voltaje en el entorno MATLAB/Simulink

Figura 34. IEDs de medición y protección

El siguiente paso en el desarrollo del modelo del sistema eléctrico es la representación de las líneas de distribución eléctrica. Estas líneas están compuestas por conductores que transportan el suministro eléctrico desde la subestación hasta los transformadores de potencia de sector, y de allí hasta cada uno de los clientes residenciales e industriales (Ver figura 36). Estas líneas deben ser capaces de permitir el paso de la corriente nominal en máxima demanda de energía en los alimentadores de distribución. El calibre del conductor que se encuentra instalado en la troncal de los alimentadores de la subestación Libertad es de 4/0 AWG ACAR (*Aluminium Conductor Alloy Reinforced*), este tipo de conductor es ideal para minimizar las pérdidas de energía y soportar las condiciones ambientales adversas.

En el entorno de simulación, para modelar este elemento de la red eléctrica, se utiliza el bloque *Three Phase Pi Section Line.* Este bloque permite simular el comportamiento de una línea de transmisión y de distribución, proporcionando así una representación realista de las características físicas (longitud de la línea) y eléctricas (frecuencia, resistencia, inductancia, capacitancia), tal como se muestra en la Figura 35.

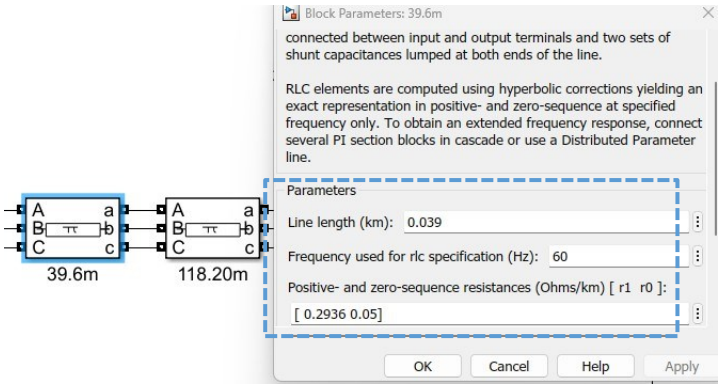


Figura 35. Configuración bloque Three Phase Section Line

Figura 36. Líneas de distribución de media tensión (13.8kV)

Luego de implementar el bloque que simula las líneas de distribución, se procede a añadir los bloques correspondientes a los transformadores de distribución y las respectivas cargas eléctricas conectadas a lo largo del alimentador de energía (Ver figura 39). Para ello, se realizó un levantamiento de información sobre la ubicación de estos transformadores, lo que permitió ingresarlos al entorno de simulación.

Dependiendo de la capacidad del transformador de cada sector del alimentador en estudio, se configura el respectivo bloque con la capacidad de potencia correspondiente y la potencia de consumo de la carga. Esta potencia de consumo se ajusta de acuerdo con las condiciones de carga que se simulen (Ver figura 38). En este caso, se ingresarán diferentes condiciones de demanda eléctrica con niveles de amperaje del 25%, 50% y 80%, lo que permitirá adquirir diferentes formas de onda en normal operación y de falla de la corriente eléctrica. Las capacidades de los transformadores de sector instalados en la red de distribución del alimentador comúnmente son de potencia de 10kVA, 15kVA, 25kVA, 37.5kVA y 50kVA, los mismos que se simulan mediante bloques en la herramienta Simulink (Ver figura 37).

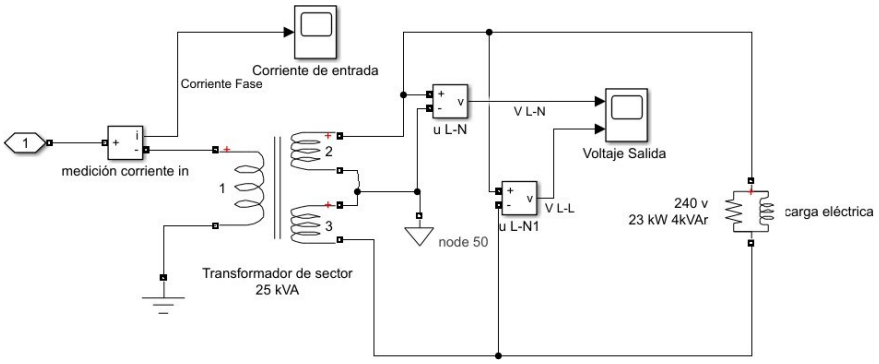


Figura 37. Bloques de simulación, transformador de distribución

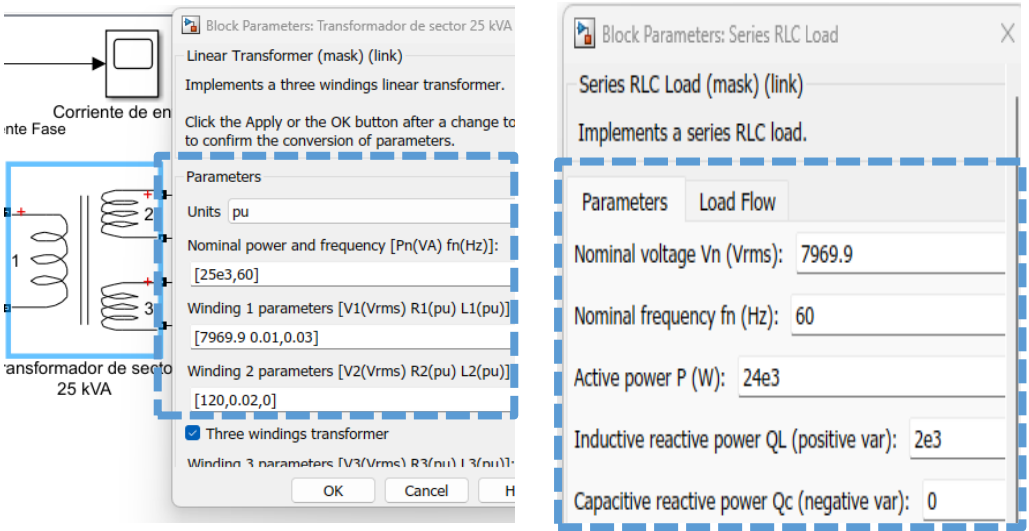


Figura 38. Configuración del bloque de transformador lineal y cargas eléctricas

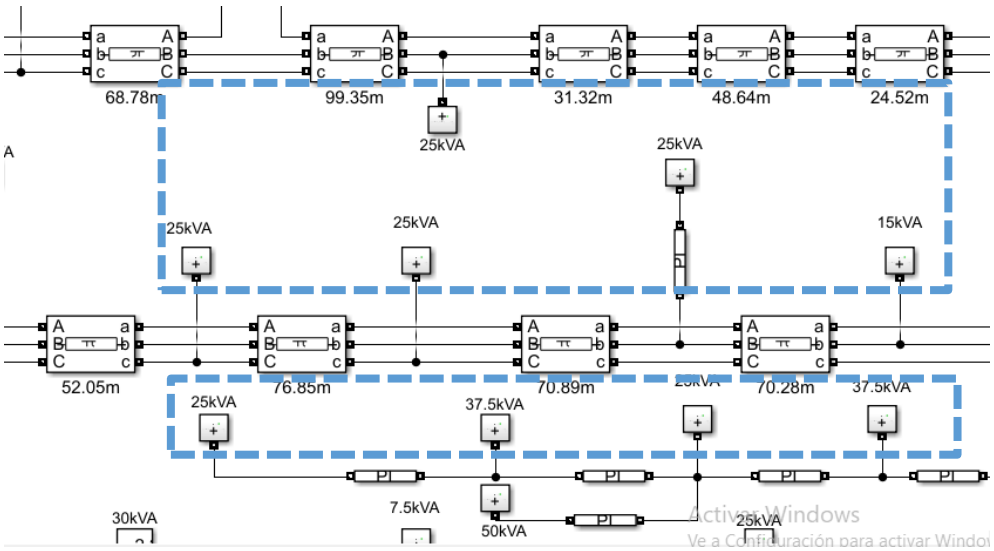


Figura 39. Simulación de cargas conectadas en el alimentador de distribución

Figura 40. Transformadores de distribución de sector

El modelo construido en la herramienta MATLAB/Simulink utilizado para la simulación y adquisición de datos de condiciones de nominal operación y de presencia de falla de alta impedancia en la onda de la corriente eléctrica, se puede visualizar a detalle en el Anexo 1, en donde se ilustra el sistema eléctrico de distribución del respectivo alimentador seleccionado, con los bloques de simulación empleados en esta sección 3.3.

3.4 Desarrollo de los modelos de inteligencia artificial

Para el desarrollo del sistema de predicción de fallas mediante IA, se llevó a cabo un procedimiento estructurado, como se muestra en la figura 41. Este procedimiento consta de cinco pasos fundamentales, donde cada uno es determinante y de vital importancia para garantizar la eficacia y precisión de los modelos de redes neuronales.

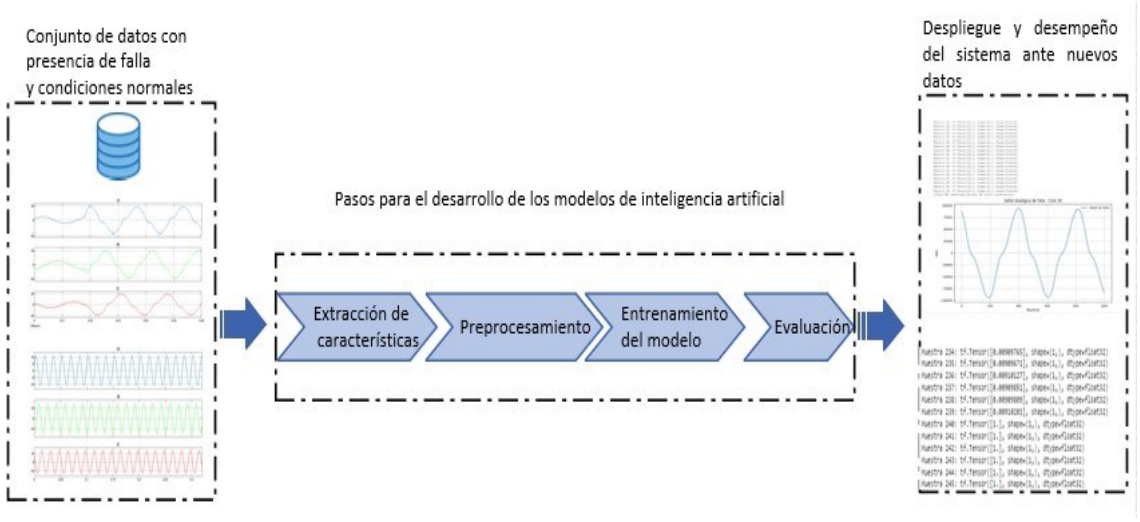


Figura 41. Estructura del desarrollo de los modelos de inteligencia artificial en la propuesta

3.4.1 Extracción de características

Durante la simulación del sistema eléctrico de distribución de un alimentador de la subestación en estudio bajo diferentes condiciones, se adquieren las ondas sinusoidales de las variables eléctricas de voltaje y corriente. Cada punto de estas señales eléctricas, muestreadas a 500 muestras por ciclo, se almacenan en estructuras de datos dentro del software MATLAB/Simulink.

De los escenarios de fallas planteados en el apartado 2.5, se pueden visualizar las siguientes ondas sinusoidales de voltaje y corriente, en diferentes superficies de contacto y condiciones de capacidad de carga al 25, 50, y 80%. Estas condiciones reflejan la variación en la demanda de amperaje que el alimentador puede soportar, siendo el 80% la máxima capacidad de demanda de amperaje registrada.

Al simular un evento de falla de alta impedancia en las fases del alimentador, se puede visualizar en la figura 42, que las ondas sinusoidales del voltaje entre líneas presentan

deformaciones periódicas. Estas deformaciones indican una alteración en el comportamiento normal del sistema eléctrico. Por ejemplo, si la falla se presenta en la fase A, se crea un desbalance en el sistema trifásico, lo que provocara que la fase C presente distorsiones debido a la interconexión entre las fases y la necesidad de mantener en equilibrio la sumatoria de voltajes en el sistema. Además, se presenta una caída en el nivel de voltaje debido a los cambios en la impedancia que se presentan durante una falla de alta impedancia (Ver figura 43).

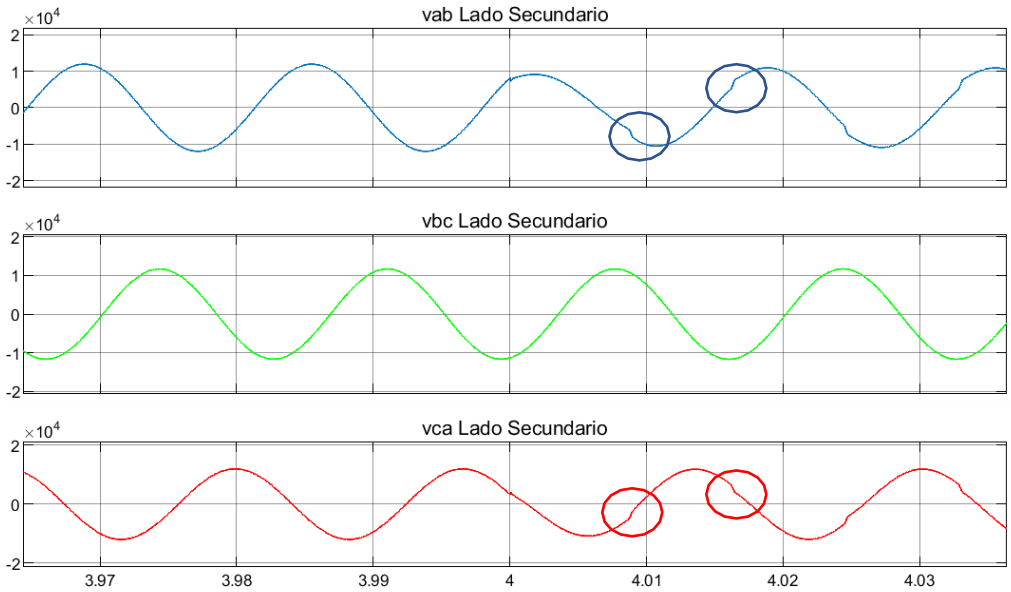


Figura 42. Voltajes de línea de pre falla y falla al simular un evento de alta impedancia en la fase A del alimentador.

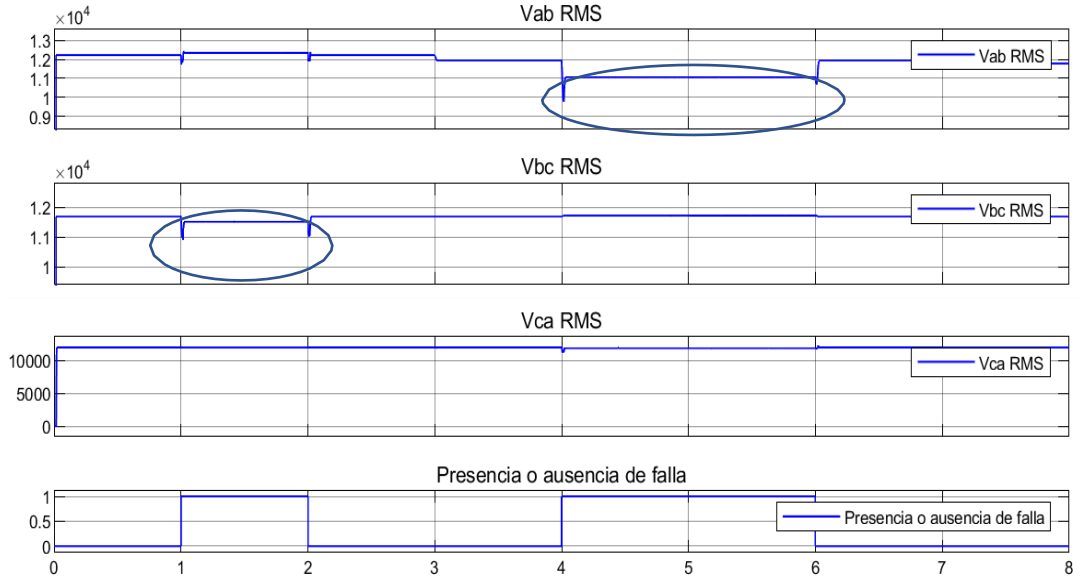


Figura 43. Nivel de voltaje eficaz en condiciones normales y de falla

En la siguiente figura se puede visualizar las corrientes de fases del sistema trifásico, en esta la fase B muestra una forma de onda característica de una falla de alta impedancia, de la cual se puede extraer las siguientes características:

Asimetría: Se muestra en la figura 45, que el pico positivo y negativo de la onda de corriente son diferentes al presentarse un evento de alta impedancia.

Forma de onda (Ver figura 44): La forma de onda de la fase B se distorsiona notablemente en comparación con la onda de corriente de las fases A y C. Esta distorsión es provocada por elementos no conductores como ramas de árboles, pastos, hormigón, entre otras superficies, que introducen no linealidades y armónicos en la onda de corriente.

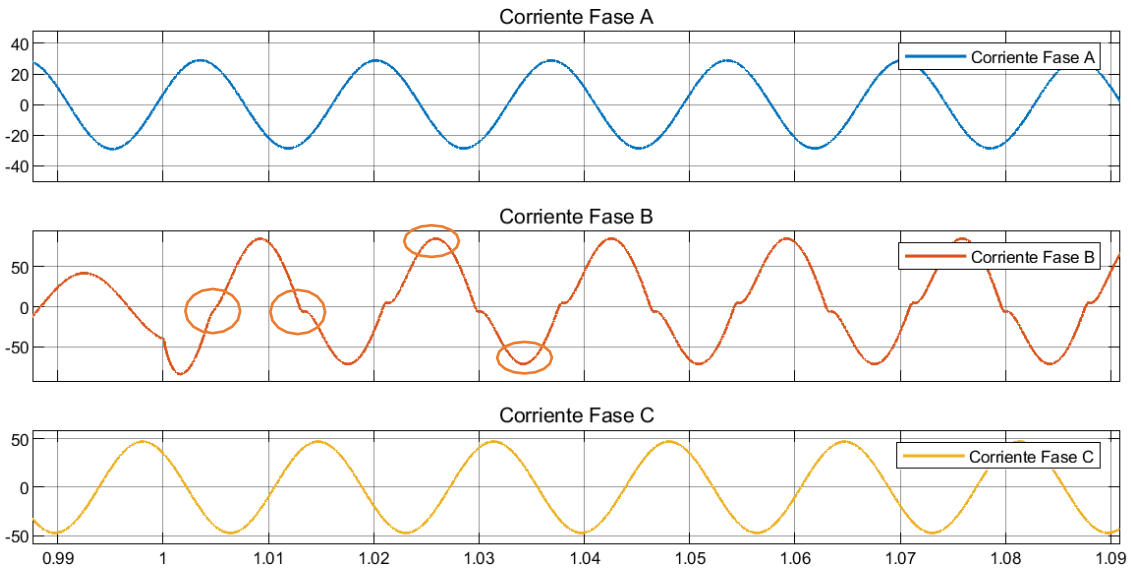


Figura 44. Corrientes de pre falla y falla de alta impedancia (Superficie: rama de árbol, Capacidad de carga del alimentador: 25%)

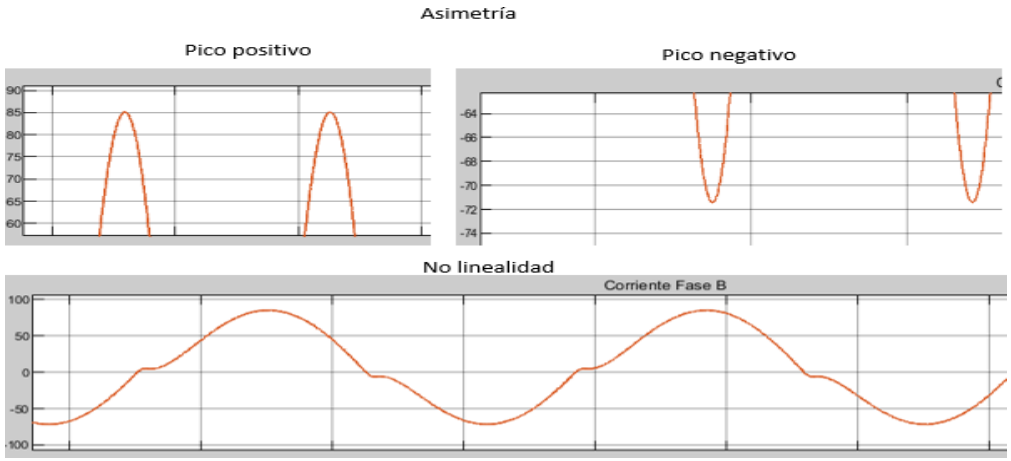


Figura 45. Características de asimetría y no linealidad en una falla de alta impedancia

El valor de la corriente eficaz al presentarse un evento de alta impedancia presenta un aumento de amperaje en su magnitud, dicho aumento bordea entre los 1 a 100 A dependiendo de la naturaleza y superficie de contacto del objeto no conductor (Ver Tabla 1). En el escenario simulado donde se modelo el contacto de la línea con varias superficies no conductoras, incluyendo pasto seco y mojado, se visualiza un aumento de 55 y 22 amperios aproximadamente en la fase A y B respectivamente (Ver figura 46). Estas corrientes de fallas son muy bajas y no son suficientes para provocar un disparo que aísle la zona de la falla, esto se debe a que las superficies presentan una alta resistencia y baja conductividad, lo que limita el flujo de corriente, dando como resultado una baja corriente de falla.

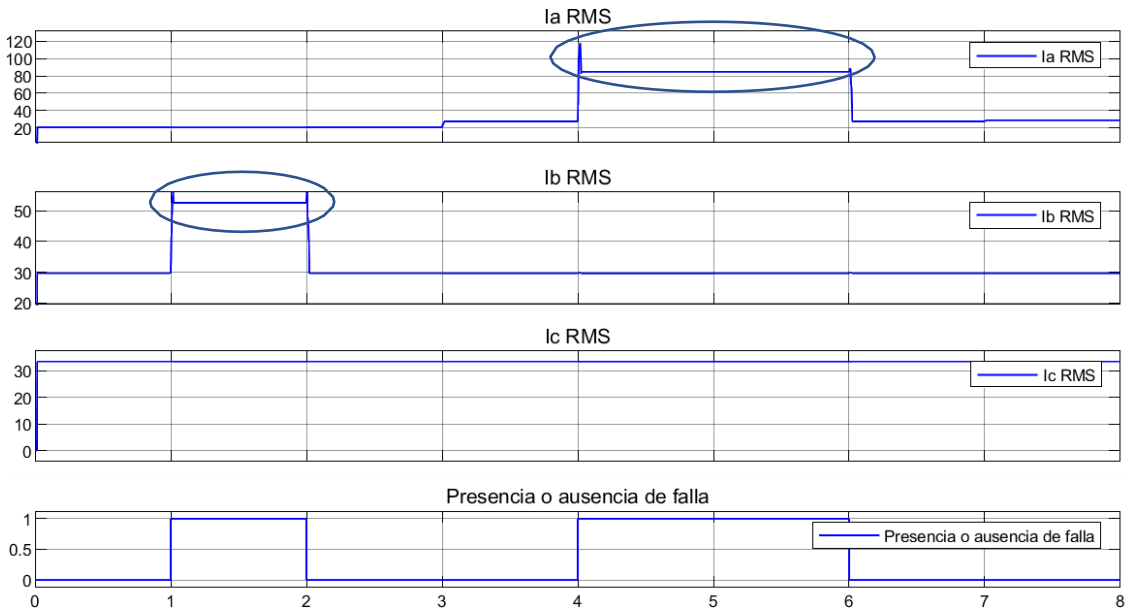


Figura 46. Valor RMS en condiciones de pre falla y falla de alta impedancia

De las características analizadas en las figuras anteriores, se opta por considerar la onda de corriente, ya que presenta las características relevantes de asimetría y distorsiones en la forma de onda, las cuales son notorias en una falla de alta impedancia, lo cual puede ayudar a la detección de las fallas en descripción. Con estas características, y mediante la recolección de datos de las simulaciones realizadas con las diferentes superficies no conductoras de alta impedancia, se forma el conjunto de datos para el entrenamiento de las redes neuronales.

3.4.2 Preprocesamiento de datos

En esta etapa, la onda de corriente en condiciones normales y condiciones de falla es segmentada en ventanas, permitiendo capturar las características y patrones asociados a las fallas de alta impedancia. Este procedimiento incrementa la cantidad de datos disponibles para el entrenamiento del modelo, lo que puede mejorar la capacidad del modelo para generalizar y detectar fallas con mayor precisión. Dado que la onda es sinusoidal con una frecuencia de 60Hz, el tamaño de la ventana se establece en dos ciclos, ya que en el apartado 3.4.1 se observó que las características de asimetría y no linealidad son periódicas durante la falla. Además, se ingresa un traslape entre ventanas equivalente a un ciclo, permitiendo así monitorear el estado anterior y el actual de la corriente eléctrica.

Las figuras 47 y 48 muestran el primer paso del preprocesamiento de datos, el cual implica un enfoque de ventana deslizante aplicado las señales sinusoidales de las fases A, B y C de corriente nominal y de falla. Este enfoque permite obtener una representación adecuada de los datos en forma de ventanas. Cada ventana tiene una duración de 2 ciclos de la onda sinusoidal de corriente y contiene 1000 datos o características (con una frecuencia de muestreo 500 muestras por ciclo). Dado que una HIF provoca pequeñas distorsiones en la forma de onda de la corriente eléctrica, contar con una gran cantidad de características mejora las capacidades de generalización y detección del sistema de IA, lo que permite obtener un sistema robusto.

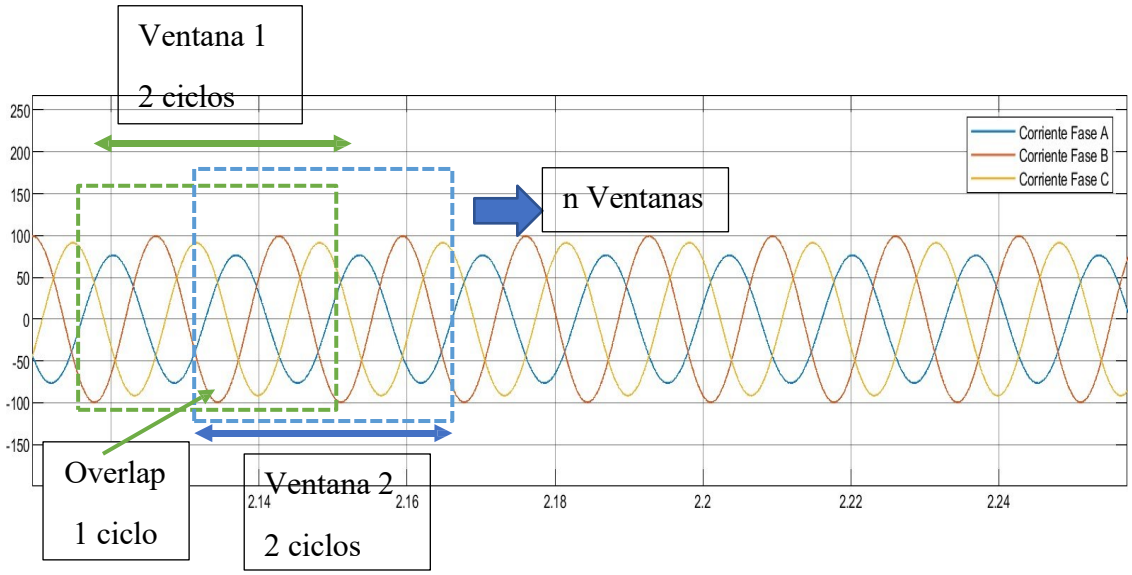


Figura 47. Enfoque de ventana deslizante, corrientes de fases de normal operación

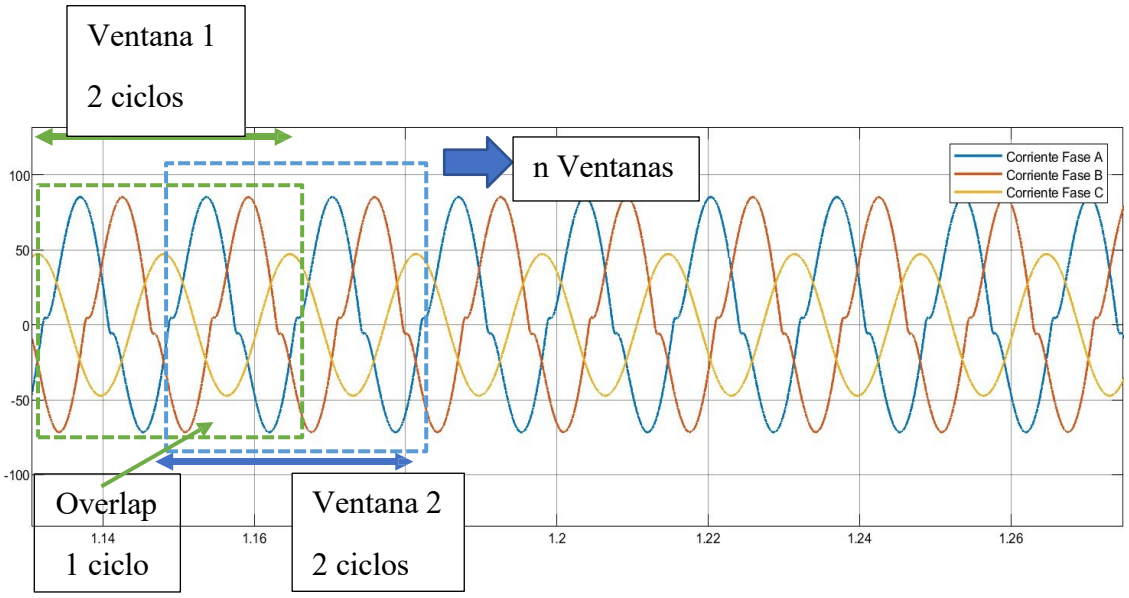


Figura 48. Enfoque de ventana deslizante, corrientes de fases con falla de alta impedancia

Cada una de las señales de corriente de las respectivas simulaciones ejecutadas se almacenan en una estructura de datos denominada *out* en el espacio de trabajo MATLAB. Para ello, se crea el código que se muestra en la figura 49, en el cual se extraen las muestras de datos y se almacena en una variable. Luego, mediante un ciclo *for,* se realizan iteraciones que permiten crear las ventanas de datos de 1000 características cada una, en la que se obtiene una matriz de datos que contiene el número de ventana, la muestra de la señal analógica de corriente con el respectivo traslape entre ventanas cada 500 muestras (Ver figura 50). Este procesa se realiza en cada simulación, considerando los diferentes tipos de superficie de alta impedancia y las diferentes fases del sistema trifásico.

```
%Separación de datos de la estructura
IA=out.IA.signals.values;%variable que contiene las muestras de datos de corriente
%AGREGAR EL OVERLAP A LOS DATOS DE LA CORRIENTE FASE A
% Definir el tamaño de la ventana y el solapamiento
tamano_ventana = 1000;%1000 muestras dos ciclos
solapamiento = 500; %500 muestras por ciclo
% Calcular la cantidad de ventanas que se crearán
num_ventanas = floor((length(IA) - tamano_ventana) / solapamiento) + 1;
% Inicializar la matriz para almacenar las ventanas
ventanas_IA = zeros(num_ventanas * tamano_ventana, 1);
% Generar las ventanas con el solapamiento requerido
for i = 1:num_ventanas
    indice_inicio = (i - 1) * solapamiento + 1;
    indice_fin = indice_inicio + tamano_ventana - 1;
    ventanas_IA((i - 1) * tamano_ventana + 1:i * tamano_ventana) = IA(indice_inicio:indice_fin);
end
% Etiquetar cada ventana
etiquetas_ventanas = repelem((1:num_ventanas)', tamano_ventana, 1);
matriz_DATA_IA= [etiquetas_ventanas, ventanas_IA];
IA_MAGNITUD=[ventanas_IA];%vector columna 999000x1
```

Figura 49. Codificación en MATLAB para crear las ventanas de datos

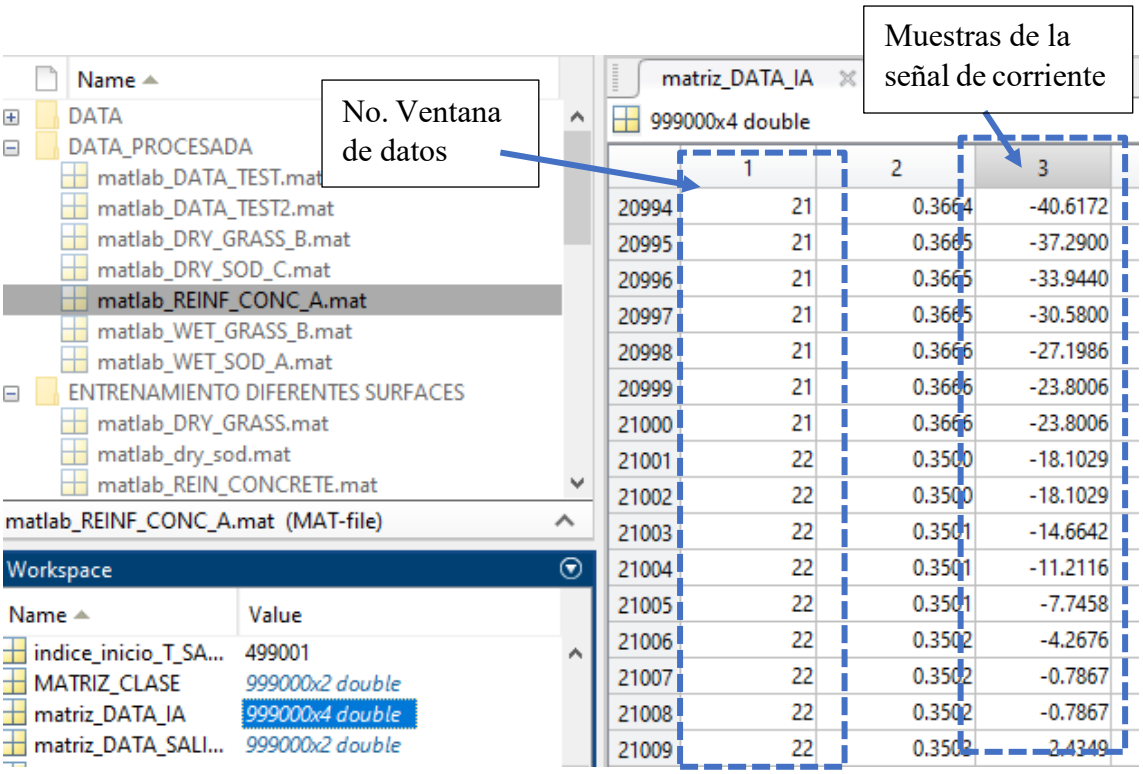


Ilustración 50. Matriz de datos que contiene el número de ventana y las muestras de corriente

Una vez obtenido el vector columna de 999000x1 datos, se toma el vector IA_MAGNITUD para reorganizarlo en ventanas de datos de 1000 características, obteniendo así un tensor de dimensión 999x1000 (Ver figura 51). En esta nueva representación, cada fila corresponde a dos ciclos de la señal sinusoidal de corriente la cual, conteniendo muestras tanto en condición nominal como en condición de falla. Posteriormente, este tensor se guarda en un archivo CSV, el cual servirá como entrada de datos para el entrenamiento de la red neuronal (Ver figura 52).

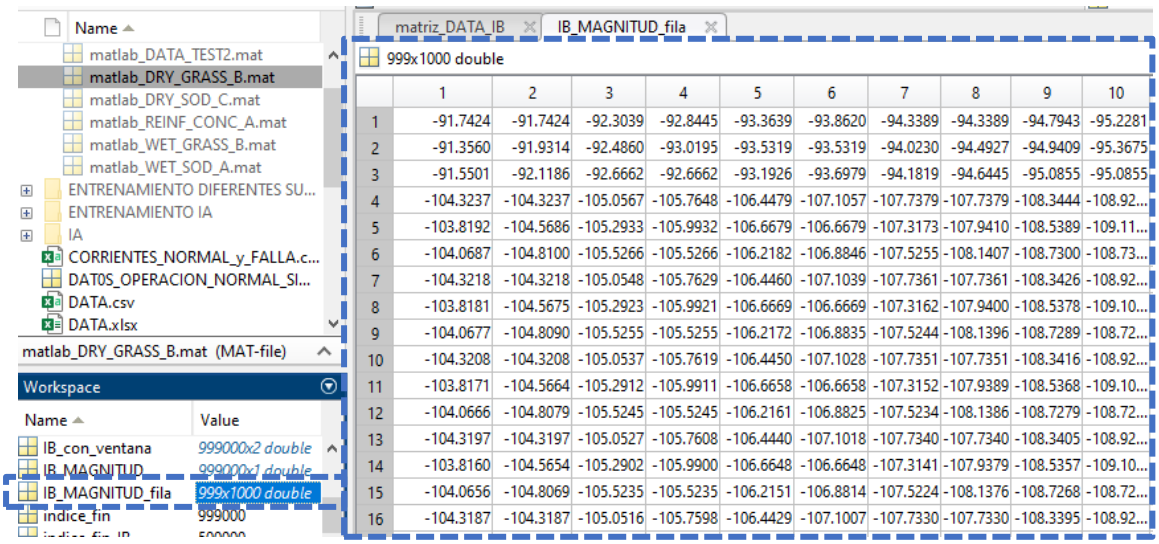
```
%REALIZAR EL VECTOR DE 999x1000
% Calcular el número de filas necesarias
num_filas = numel(IA_MAGNITUD) / 1000;
% Inicializar un nuevo vector fila
IA_MAGNITUD_fila = zeros(num_filas, 1000);
% Recorrer el vector columna y llenar el vector fila
for i = 1:num_filas
    indice_inicio = (i - 1) * 1000 + 1;
    indice_fin = i * 1000;
    IA_MAGNITUD_fila(i, :) = IA_MAGNITUD(indice_inicio:indice_fin)';
end
```

Figura 51. Codificación en MATLAB para crear las ventanas de datos

```
% Nombre el archivo CSV
nombre_archivo = 'CORRIENTES_NORMAL_y_FALLA_REIN_CONC_FASE_A.csv';
% Directorio de salida
directorio_salida = 'C:\Users\OneDrive\Escritorio\SIMULACION\DATA ENTRENAMIENTO IA\DATA';
% Ruta completa del archivo CSV
ruta_completa = fullfile(directorio_salida, nombre_archivo);
% Guardar la matriz en el archivo CSV usando csvwrite
csvwrite(ruta_completa, IA_MAGNITUD_fila);
```

Figura 52. Codificación en MATLAB para guardar el conjunto de datos en un archivo CSV

La figura 53 muestra un bloque de datos procesado en el software MATLAB el cual contiene las muestras de condiciones normales y fallas de alta impedancia. Este bloque de datos almacenado en la variable tipo vector "IB_MAGNITUD_fila" de dimensión 999x1000, contiene 999 ventanas y 1000 características, entre ellas asimetrías y no linealidad de la onda de falla de alta impedancia. De manera similar, se construyen otros conjuntos de datos con diferentes superficies de fallas y condiciones de carga (Ver Tabla 8) procesados de manera vectorial, con el fin de obtener un amplio conjunto de datos y características para el entrenamiento de la red neuronal.



	1	2	3	4	5	6	7	8	9	10
1	-91.7424	-91.7424	-92.3039	-92.8445	-93.3639	-93.8620	-94.3389	-94.3389	-94.7943	-95.2281
2	-91.3560	-91.9314	-92.4860	-93.0195	-93.5319	-93.5319	-94.0230	-94.4927	-94.9409	-95.3675
3	-91.5501	-92.1186	-92.6662	-92.6662	-93.1926	-93.6979	-94.1819	-94.6445	-95.0855	-95.0855
4	-104.3237	-104.3237	-105.0567	-105.7648	-106.4479	-107.1057	-107.7379	-107.7379	-108.3444	-108.92...
5	-103.8192	-104.5686	-105.2933	-105.9932	-106.6679	-106.6679	-107.3173	-107.9410	-108.5389	-109.11...
6	-104.0687	-104.8100	-105.5266	-105.5266	-106.2182	-106.8846	-107.5255	-108.1407	-108.7300	-108.73...
7	-104.3218	-104.3218	-105.0548	-105.7629	-106.4460	-107.1039	-107.7361	-107.7361	-108.3426	-108.92...
8	-103.8181	-104.5675	-105.2923	-105.9921	-106.6669	-106.6669	-107.3162	-107.9400	-108.5378	-109.10...
9	-104.0677	-104.8090	-105.5255	-105.5255	-106.2172	-106.8835	-107.5244	-108.1396	-108.7289	-108.72...
10	-104.3208	-104.3208	-105.0537	-105.7619	-106.4450	-107.1028	-107.7351	-107.7351	-108.3416	-108.92...
11	-103.8171	-104.5664	-105.2912	-105.9911	-106.6658	-106.6658	-107.3152	-107.9389	-108.5368	-109.10...
12	-104.0666	-104.8079	-105.5245	-105.5245	-106.2161	-106.8825	-107.5234	-108.1386	-108.7279	-108.72...
13	-104.3197	-104.3197	-105.0527	-105.7608	-106.4440	-107.1018	-107.7340	-107.7340	-108.3405	-108.92...
14	-103.8160	-104.5654	-105.2902	-105.9900	-106.6648	-106.6648	-107.3141	-107.9379	-108.5357	-109.10...
15	-104.0656	-104.8069	-105.5235	-105.5235	-106.2151	-106.8814	-107.5224	-108.1376	-108.7268	-108.72...
16	-104.3187	-104.3187	-105.0516	-105.7598	-106.4429	-107.1007	-107.7330	-107.7330	-108.3395	-108.92...

Figura 53. Matriz de un conjunto de datos de operación nominal y de falla

Las simulaciones se realizan mediante 1000 ciclos, lo que corresponde a 16.6s de simulación del sistema eléctrico en MATLAB/Simulink. En estas simulaciones, se modelan fallas de las diferentes superficies de alta impedancia en varios instantes de tiempo, asegurando un equilibrio entre los datos normales y de falla que se utilizarán para el entrenamiento de la red neuronal. De esta manera, al entrenar el sistema de inteligencia artificial, se pueda obtener un modelo capaz de clasificar correctamente las diferentes condiciones del sistema eléctrico (presencia de falla o normal operación).

Tabla 8. Conjuntos de datos para el entrenamiento del sistema de inteligencia artificial

Conjunto de datos para el entrenamiento de la IA				
Superficie no conductora	**No. Ventanas sin falla**	**No. Ventanas con falla**	**Condición de carga**	**Fase falla**
rama de árbol	520	480	80%	C
césped seco	520	480	50%	A
pasto seco	520	480	50%	A
césped húmedo	520	480	25%	B
pasto mojado	520	480	25%	B
hormigón reforzado	520	480	25%	C
Número de ventanas	3120	2880		
Número de muestras	3120000	2880000		

Posteriormente, se realiza la codificación de las salidas del modelo de inteligencia artificial para clasificar los estados del sistema. En este caso, se definen dos variables categóricas: ausencia y presencia de falla de alta impedancia. Estas variables se codifican mediante clases, donde la clase 0 representa la ausencia de falla y la clase 1 indica la detección de una falla de alta impedancia (Ver figura 54). Esta codificación binaria permite una interpretación clara y directa de los resultados, al evaluar el modelo de inteligencia artificial, lo que facilita una respuesta rápida y precisa en el monitoreo y diagnóstico del sistema eléctrico.

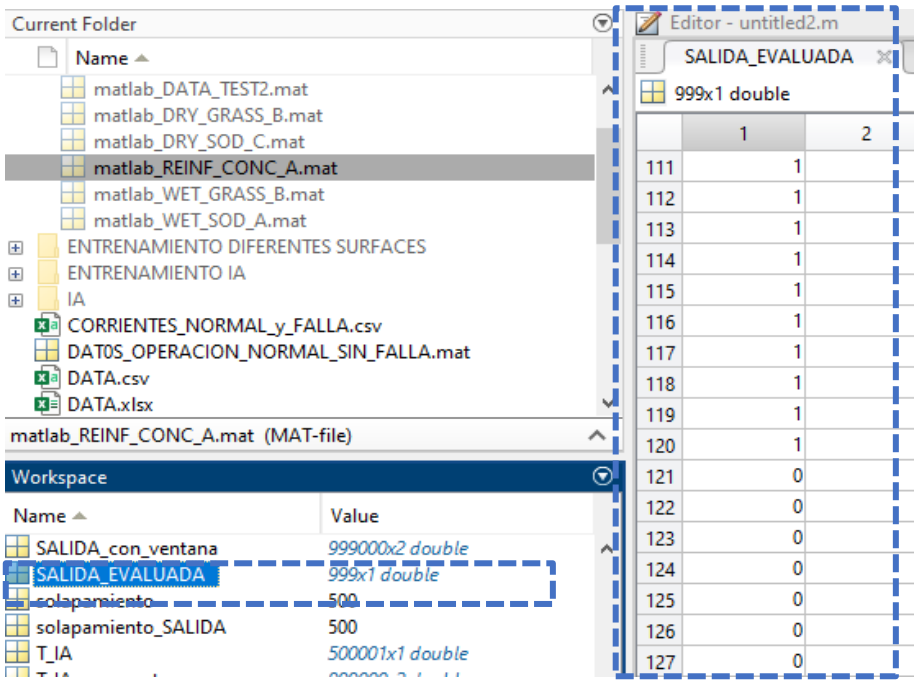


Figura 54. Codificación de la salida de la inteligencia artificial mediante clases 0 y 1

Para garantizar la precisión del modelo, se aplica una técnica de balance de datos para garantizar que ambas clases estén representadas de manera equitativa en el conjunto de datos de la tabla 8. Esto previene sesgos en el modelo y mejora su capacidad para detectar fallas con precisión. Además, el conjunto de datos se divide en un 80% para entrenamiento y un 20% para prueba, permitiendo evaluar el rendimiento del modelo de manera objetiva y asegurando que generalice bien a nuevos datos no vistos durante el entrenamiento.

En la figura 55 se muestra el proceso de carga de datos en el entorno de Google Colab y lenguaje Python correspondiente a los datos muestreados de la onda de corriente en condiciones normal y de falla en conjunto con los datos etiquetados de salida el cual contiene las respectivas clases, todos almacenados en archivos CSV. Posteriormente, los datos de corriente y las etiquetas de salida se dividen en conjuntos de entrenamiento y prueba utilizando la función train_test_split. Esta función es ampliamente utilizada en el preprocesamiento de datos para modelos de aprendizaje automático.

Figura 55. División del conjunto de datos en un 80% para entrenamiento y 20% para prueba

3.4.3 Modelos de inteligencia artificial propuestos

En el presente trabajo se ha propuesto varios modelos de aprendizaje automático y redes neuronales con diferentes arquitecturas para evaluar la detección de fallas de alta impedancia en sistemas eléctricos, tales como *Multiperceptron Layer*, Redes convolucionales, los cuales en los estudios realizados en (Fan & Yin, 2019), (Wang & Chen, 2019) alcanzaron una precisión de detección por encima del 99%. En base a la investigación realizada en (Moloi et al., 2018), se plantea de igual forma un modelo SVM, ya que son precisos en problemas de clasificación binaria. En dicha investigación el porcentaje de precisión en la detección de este tipo de fallas se situó por encima del 95%, demostrando la capacidad de los modelos de inteligencia artificial para reconocer

patrones y clasificar fallas en un sistema eléctrico. Los modelos son desarrollados en el entorno de Google Colab utilizando *frameworks* como Tensorflow y Keras en el lenguaje de programación Python. A continuación, se describen las arquitecturas:

3.4.3.1 MPL

Se propone dos modelos de red neuronal *Multiperceptron Layer* de siete y nueve capas. Estas redes consisten en capas densas totalmente conectadas, donde, cada neurona de una capa se conecta con todas las neuronas de la capa siguiente. Una red MPL es adecuada para datos tabulares, secuenciales, y series de tiempo aplicando la técnica de ventana deslizante. Esta técnica es utilizada en esta propuesta para crear conjuntos de características a partir de los datos secuenciales, permite que la MPL aprenda relaciones no lineales entre los datos de entrada y las etiquetas de salida, captando patrones o características sutiles como la asimetría en la onda de corriente en una falla de alta impedancia. Las arquitecturas de las MPL utilizadas se muestran en las tablas 9 y 10, en las que se varían el número de capas densas conectadas.

Tabla 9. Arquitectura 1 Red MPL

Arquitectura 1 Red MPL					
Capas	**Tamaño datos entrada**	**Tipo de Capa**	**Neuronas**	**Función activación**	**Regularización**
7	1000	Capa 1 oculta	512	ReLU	L2
		Capa 2 oculta	256	ReLU	L2
		Capa 3 oculta	128	ReLU	L2
		Capa 4 oculta	64	ReLU	L2
		Capa 5 oculta	32	ReLU	L2
		Capa 6 oculta	16	ReLU	L2
		Capa de salida	1	sigmoide	N/A

Tabla 10. Arquitectura2 Red MPL

Arquitectura 2 Red MPL					
Capas	**Tamaño datos entrada**	**Tipo de Capa**	**Neuronas**	**Función activación**	**Regularización**
9	1000	Capa 1 oculta	2048	ReLU	L2
		Capa 2 oculta	1024	ReLU	L2
		Capa 3 oculta	512	ReLU	L2
		Capa 4 oculta	256	ReLU	L2
		Capa 5 oculta	128	ReLU	L2

		Capa 6 oculta	64	ReLU	L2
		Capa 7 oculta	32	ReLU	L2
		Capa 8 oculta	16	ReLU	L2
		Capa de salida	1	sigmoide	N/A

Las figuras 56 y 57 muestran la implementación de las dos arquitecturas MPL propuestas en el entorno de Colab, utilizando la instrucción *Sequential*(), que define un modelo secuencial en Keras compuesto por una serie de capas totalmente conectadas. Las arquitecturas planteadas varían entre siete y nueve capas para hacer la red más profunda. En estas arquitecturas propuestas se utiliza la función de activación ReLU en las capas densas internas, ya que permite aprender patrones complejos y, además, es una función simple que alivia el consumo de recursos computacionales durante el entrenamiento. Para la capa de salida se usa la función de activación sigmoidal, ya que su salida está acotada entre 0 y 1, lo que la hace ideal para aplicaciones de clasificación binaria, tal como es el caso de la predicción de ausencia o presencia de fallas. Además, se emplea la regularización L2 para prevenir el sobreajuste y mejorar la generalización del modelo ante nuevos datos.

```
model = Sequential([
    Dense(512, activation='relu', input_shape=(1000,), kernel_regularizer='l2'),
    Dense(256, activation='relu', kernel_regularizer='l2'),
    Dense(128, activation='relu', kernel_regularizer='l2'),
    Dense(64, activation='relu', kernel_regularizer='l2'),
    Dense(32, activation='relu', kernel_regularizer='l2'),
    Dense(16, activation='relu', kernel_regularizer='l2'),
    Dense(1, activation='sigmoid')
])
```

Figura 56. Implementación de la primera arquitectura MPL

```
model = Sequential([
    Dense(2048, activation='relu', input_shape=(1000,), kernel_regularizer='l2'),
    Dense(1024, activation='relu', kernel_regularizer='l2'),
    Dense(512, activation='relu', kernel_regularizer='l2'),
    Dense(256, activation='relu', kernel_regularizer='l2'),
    Dense(128, activation='relu', kernel_regularizer='l2'),
    Dense(64, activation='relu', kernel_regularizer='l2'),
    Dense(32, activation='relu', kernel_regularizer='l2'),
    Dense(16, activation='relu', kernel_regularizer='l2'),
    Dense(1, activation='sigmoid')
])
```

Figura 57. Implementación de la segunda arquitectura MPL

3.4.3.2 CNN

Conjuntamente para abordar esta tarea de detectar fallas de alta impedancia en sistemas eléctricos de potencia, se propone dos arquitecturas avanzadas de redes neuronales convolucionales (Ver Tabla 11 y 12). La elección de estas arquitecturas se fundamenta en la capacidad que tienen las CNN para capturar patrones complejos en datos secuenciales, como las señales eléctricas. La primera arquitectura, compuesta por siete capas, ofrece una estructura más simplificada y rápida de entrenar, la cual es adecuada en primera instancia para obtener una respuesta ágil del desempeño de la arquitectura ante los datos de entrenamiento ingresados. En contraste, la segunda red neuronal, con doce capas, está diseñada de manera profunda y robusta, lo cual permitirá un análisis más detallado y una mayor capacidad de generalización. Ambas redes están configuradas para una salida binaria donde una clase 0 indica una operación normal y una clase 1 señala la presencia de una falla eléctrica.

Tabla 11. Arquitectura 3 Red CNN

Arquitectura 3 Red CNN					
Capas	**Tamaño datos entrada**	**Tipo de capa**	**Kernels/ filtros**	**Neuronas**	**Función de activación**
7	1000	Capa 1 Convolucional 1D	64	N/A	N/A
		Capa 2 Convolucional 1D	128	N/A	N/A
		Capa 3 Max Pooling	N/A	N/A	N/A
		Capa 4 Flatten	N/A	N/A	N/A
		Capa 5 densa (totalmente conectada)	N/A	128	ReLU
		Capa 6 densa (totalmente conectada)	N/A	64	ReLU
		Capa 7 de salida	N/A	1	sigmoide

Tabla 12. Arquitectura 4 Red CNN

Arquitectura 4 Red CNN					
Capas	**Tamaño datos entrada**	**Tipo de capa**	**Kernels/ filtros**	**Neuronas**	**Función de activación**
12	1000	Capa 1 Convolucional 1D	64	N/A	ReLU
		Capa 2 MaxPooling 1D	N/A	N/A	N/A
		Capa 3 Convolucional 1D	128	N/A	ReLU
		Capa 4 MaxPooling 1D	N/A	N/A	N/A
		Capa 5 Convolucional 1D	256	N/A	ReLU
		Capa 6 MaxPooling 1D	N/A	N/A	N/A
		Capa 7 Convolucional 1D	512	N/A	ReLU
		Capa 8 MaxPooling 1D	N/A	N/A	N/A
		Capa 9 Flatten	N/A	N/A	N/A
		Capa 10 densa (totalmente conectada)	N/A	1024	ReLU
		Capa 11 densa (totalmente conectada)	N/A	512	ReLU
		Capa 12 de salida	N/A	1	sigmoide

Las figuras 58 y 59 muestran la implementación de dos modelos de red neuronal convolucional en el entorno de google Colab. El primer modelo, compuesto por 8 capas, se construye utilizando la instrucción *Sequential*. A través de la instrucción model.add, se incorporan las diferentes capas del modelo: 2 capas convolucionales, 1 capa de *MaxPooling*, 1 capa de aplanamiento de datos, 2 capas totalmente conectadas y finalmente una capa de salida con una función de activación sigmoide adecuada para tareas de clasificación binaria. Las capas convolucionales aplican filtros que permiten aprender características importantes de los datos de entrada, mientras que la capa de *MaxPooling* reduce la dimensionalidad, conservando las características más relevantes y ayudando a mitigar el sobreajuste. Este primer modelo de CNN está diseñado para ser computacionalmente eficiente debido a su estructura y a las capas convolucionales, que permiten extraer y aprender características jerárquicas y relevantes de las muestras de datos de la señal de corriente en condiciones de normal operación y de falla.

```
# Definir el modelo
model = Sequential()
# Agregar capas convolucionales
model.add(Conv1D(64, kernel_size=3, activation='relu', input_shape=(1000, 1)))
model.add(Conv1D(128, kernel_size=3, activation='relu'))
model.add(Conv1D(256, kernel_size=3, activation='relu'))
model.add(MaxPooling1D(pool_size=2))
# Aplanar los datos
model.add(Flatten())
# Agregar capas densas
model.add(Dense(128, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

Figura 58. Implementación de la tercera arquitectura red CNN

En el segundo modelo mostrado en la figura 59, se propone una arquitectura más profunda y compleja, lo que puede resultar beneficioso por el gran volumen de los datos de entrada que se utiliza en la propuesta. Se tiene un total de 4 capas convoluciones con filtros de 64,128,256 y 512 respectivamente, cada una de estas capas esta seguida por una capa de *MaxPooling* y al final una capa de aplanamiento de datos (*Flatten*), luego se utilizan dos capas de 1024 y 512 neuronas respectivamente con una función de activación ReLU, finalmente la capa de salida es similar a la arquitectura anterior con una neurona respectiva para realizar la detección de fallas de alta impedancia. Este modelo que se propone debido a la complejidad y profundidad de la red, se espera que tenga un comportamiento óptimo y una mayor capacidad para capturar y predecir los patrones de operación nominal y de falla de alta impedancia en una onda sinusoidal de corriente eléctrica.

```
model = Sequential()
# Primer bloque de capas convolucionales
model.add(Conv1D(64, kernel_size=3, activation='relu', input_shape=(1000, 1))
model.add(MaxPooling1D(pool_size=2))
# Segundo bloque de capas convolucionales
model.add(Conv1D(128, kernel_size=3, activation='relu'))
model.add(MaxPooling1D(pool_size=2))
# Tercer bloque de capas convolucionales
model.add(Conv1D(256, kernel_size=3, activation='relu'))
model.add(MaxPooling1D(pool_size=2))
# Cuarto bloque de capas convolucionales
model.add(Conv1D(512, kernel_size=3, activation='relu'))
model.add(MaxPooling1D(pool_size=2))
# Aplanar y agregar capas completamente conectadas
model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dense(512, activation='relu'))
# Capa de salida
model.add(Dense(1, activation='sigmoid'))
```

Figura 59. Implementación de la cuarta arquitectura red CNN

3.4.3.3 SVM

Para abordar la tarea de clasificación binaria de determinar la presencia o ausencia de falla de alta impedancia en el sistema eléctrico de la subestación La Libertad, se propone de igual forma el uso de un algoritmo de aprendizaje supervisado como lo es una Máquina de Soporte Vectorial (SVM). La SVM es utilizada principalmente para clasificación y regresión de datos. Dado que el conjunto de datos consiste en ventanas de datos de alta dimensionalidad, con 1000 características cada una, y la salida que se requiere predecir es binaria (0 o 1), lo que indica la ausencia o presencia de falla respectivamente, la SVM emerge como una opción óptima.

En este caso se propone un modelo de SVM para abordar la tarea en ejecución. Se emplea un kernel lineal con un factor de regularización C de 0.01 y un factor gamma ajustado automáticamente mediante la opción *gamma=scale* (Ver figura 60*)*. Esta configuración es ideal, ya que proporciona una fuerte regularización y una configuración inicial robusta que puede ayudar a prevenir sobreajuste, especialmente útil en escenarios donde se tiene series temporales, tal como el caso de los datos que se utilizan en esta propuesta, donde la detección de patrones de falla puede ser complicada debido a la naturaleza de la señal de falla.

```
[18] #Crear el modelo SVM
     scaler = StandardScaler()
     svm = SVC(kernel='linear', C=0.01, gamma='scale', probability=True, verbose=True)
     model = make_pipeline(scaler, svm)
```

Figura 60. Modelo de la arquitectura 5 SVM con kernel lineal

La combinación de los diferentes enfoques en los modelos de inteligencia artificial propuestos permitirá tener una visión integral de la capacidad de dichos modelos para abordar la detección de fallas de alta impedancia mediante patrones en señales analógicas de corriente eléctrica y clasificación binaria, adicionalmente la respectiva codificación de cada de unas estas arquitecturas se muestran en los Anexos 2,3,4,5, y 6.

3.4.4 Entrenamiento de las arquitecturas de inteligencia artificial propuestas

Un paso previo al entrenamiento de la red neuronal es compilar los modelos propuestos. Esto se logra utilizando la instrucción model.compile (Ver figura 61), donde se especifica el algoritmo de optimización, la función de pérdida y la lista de métricas para medir el desempeño del modelo durante el entrenamiento y la validación. En este caso, se selecciona el algoritmo de optimización Adam, que ajusta iterativamente la tasa de aprendizaje durante el entrenamiento. La función de pérdida elegida es la entropía cruzada binaria, ampliamente utilizada en problema de clasificación binaria. Además, se utiliza la métrica de precisión (*acurracy*) para monitorear el rendimiento del modelo en cada época de entrenamiento. También se mostrarán los valores de perdida de entrenamiento y validación lo que permite visualizar como progresa el proceso de aprendizaje del modelo durante el entrenamiento.

Una vez que se compila los respectivos modelos y se ha seleccionado los hiperparámetros óptimos, el modelo está listo para ser entrenado con el respectivo conjunto de datos de entrada.

```
[ ]  model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

Figura 61. Codificación para compilar el modelo de una red MPL y CNN

Luego de la selección de características, procesamiento de datos, selección y compilación de modelos de inteligencia artificial, se procede a entrenar los diferentes modelos de IA planteados. Durante este proceso los modelos aprenden a realizar predicciones correctas a partir de los datos de entrenamiento. Esta etapa es crítica, ya que a medida que avanza el entrenamiento, el modelo adquiere la capacidad de generalizar ante datos no vistos anteriormente. Además, ajusta sus parámetros internos, como los pesos y sesgos, para el caso de las redes MPL y CNN, iterativamente para minimizar las perdidas, que miden la brecha entre las predicciones y las etiquetas verdaderas de los datos de entrenamiento.

Las siguientes figuras 62, 63, 64 y 65 muestran una sección del proceso de entrenamiento durante varias épocas, en las que se observan las respectivas pérdidas y precisiones durante el entrenamiento y la validación. Cada una de las arquitecturas, MPL y CNN, es

entrenada durante 50 y 150 épocas, utilizando 4800 ventanas de datos para el entrenamiento y 1200 ventanas para validación.

Para esta propuesta, se seleccionaron dos valores de batch size: un valor de 4 para la primera arquitectura MPL y un valor de 32 para las otras tres arquitecturas. La elección de un batch size de 32 se sustenta en pruebas preliminares que mostraron que este valor permitía un entrenamiento estable y una convergencia rápida (Ver figuras 64 y 65). Además, se observó que un batch size de 32 utilizaba eficientemente los recursos de hardware sin afectar el rendimiento del modelo entrenado, lo cual es ideal para arquitecturas complejas como las CNN empleadas. También se produjeron mejores resultados en términos de precisión y pérdida en la validación, permitiendo alcanzar una buena generalización de los modelos de IA sin comprometer la velocidad de entrenamiento.

Por otro lado, un batch size pequeño, como 4, permite actualizaciones más frecuentes en los pesos de la red, pero con el riesgo de introducir ruido en el proceso de optimización. En resumen, la elección del tamaño del batch size debe ser cuidadosamente considerada, ya que influye significativamente en la dinámica del entrenamiento y en las métricas del rendimiento de los respectivos modelos de IA descritos en esta propuesta.

```
# Entrenar el modelo MPL
history = model.fit(X_train, y_train, epochs=150, batch_size=1, validation_data=(X_test, y_test))
```

```
Epoch 122/150
4795/4795 [==============================] - 44s 9ms/step - loss: 0.2017 - accuracy: 0.9472 - val_loss: 0.3947 - val_accuracy: 0.8791
Epoch 123/150
4795/4795 [==============================] - 45s 9ms/step - loss: 0.2230 - accuracy: 0.9406 - val_loss: 0.1822 - val_accuracy: 0.9558
Epoch 124/150
4795/4795 [==============================] - 45s 9ms/step - loss: 0.2279 - accuracy: 0.9351 - val_loss: 0.2508 - val_accuracy: 0.9550
Epoch 125/150
4795/4795 [==============================] - 46s 10ms/step - loss: 0.2097 - accuracy: 0.9449 - val_loss: 0.2074 - val_accuracy: 0.9608
Epoch 126/150
4795/4795 [==============================] - 45s 9ms/step - loss: 0.2220 - accuracy: 0.9414 - val_loss: 0.1501 - val_accuracy: 0.9575
Epoch 127/150
4795/4795 [==============================] - 46s 10ms/step - loss: 0.2131 - accuracy: 0.9397 - val_loss: 0.1540 - val_accuracy: 0.9566
Epoch 128/150
4795/4795 [==============================] - 45s 9ms/step - loss: 0.2170 - accuracy: 0.9404 - val_loss: 0.2504 - val_accuracy: 0.9266
Epoch 129/150
4795/4795 [==============================] - 45s 9ms/step - loss: 0.2142 - accuracy: 0.9389 - val_loss: 0.2138 - val_accuracy: 0.9391
Epoch 130/150
4795/4795 [==============================] - 45s 9ms/step - loss: 0.2195 - accuracy: 0.9439 - val_loss: 0.1770 - val_accuracy: 0.9458
Epoch 131/150
4795/4795 [==============================] - 46s 10ms/step - loss: 0.2128 - accuracy: 0.9424 - val_loss: 0.1967 - val_accuracy: 0.9558
Epoch 132/150
4795/4795 [==============================] - 47s 10ms/step - loss: 0.2270 - accuracy: 0.9370 - val_loss: 0.2589 - val_accuracy: 0.9366
Epoch 133/150
4795/4795 [==============================] - 47s 10ms/step - loss: 0.2166 - accuracy: 0.9420 - val_loss: 0.1536 - val_accuracy: 0.9967
Epoch 134/150
4795/4795 [==============================] - 47s 10ms/step - loss: 0.2097 - accuracy: 0.9397 - val_loss: 0.2383 - val_accuracy: 0.9058
```

Figura 62. Entrenamiento de la arquitectura 1 Red MPL (bath size = 4)

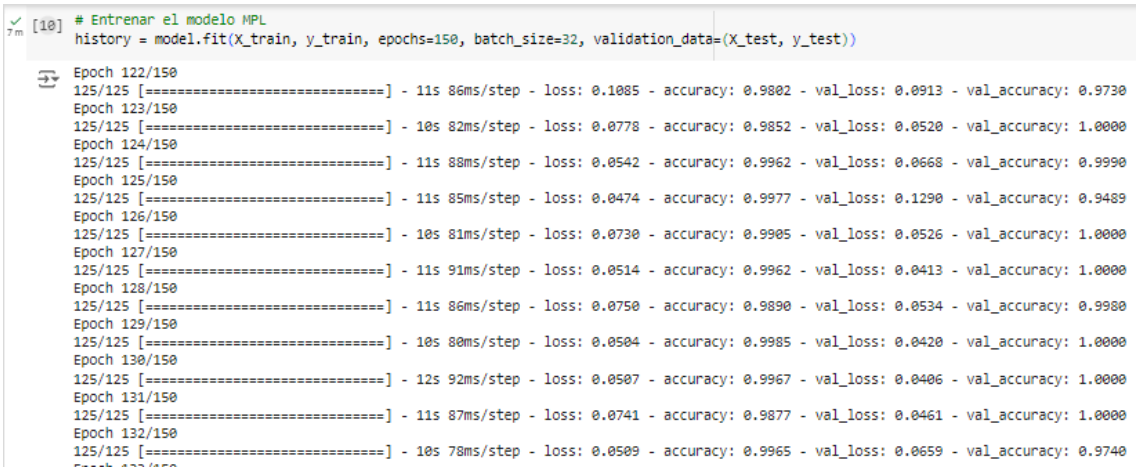


Figura 63. Entrenamiento de la arquitectura 2 Red MPL (batch size = 32)

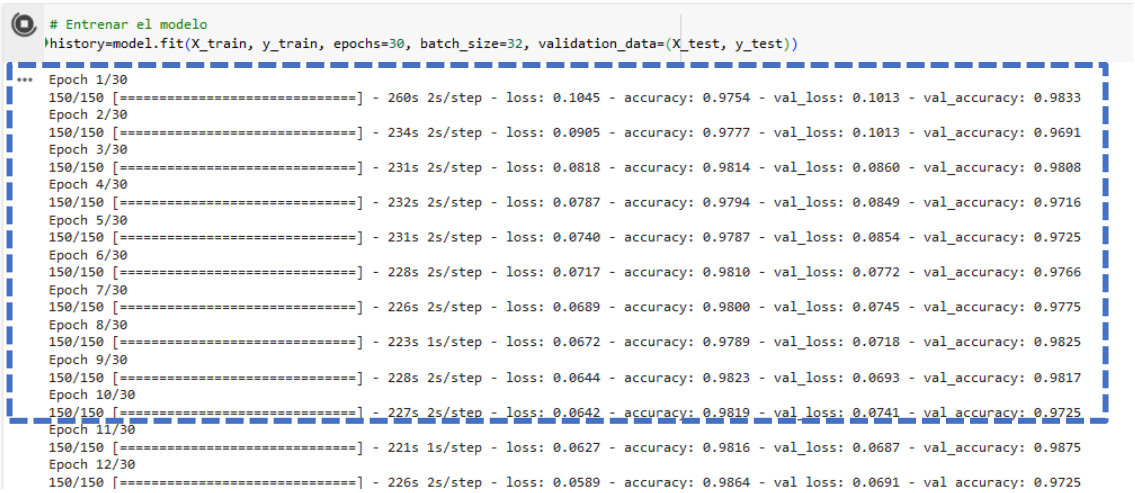


Figura 64. Entrenamiento de la arquitectura 3 Red CNN (batch size=32)

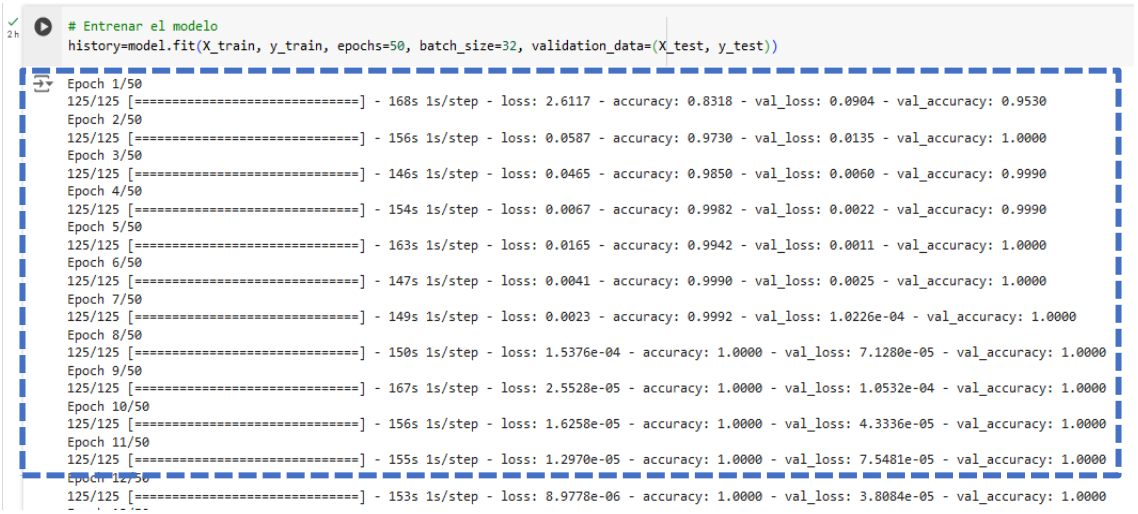


Figura 65. Entrenamiento de la arquitectura 4 Red CNN (batch size = 3

3.5 Pruebas y Resultados

3.5.1 Descripción de los datos eléctricos obtenidos del modelo de simulación desarrollado en MATLAB/Simulink

En este apartado se analiza los datos, variables eléctricas y formas de onda al simular eventos de alta impedancia en la red eléctrica. La figura 66 muestra las corrientes trifásicas del sistema de distribución eléctrico de un alimentador de la subestación La Libertad, en la que en el tiempo t=1s se produce un evento de alta impedancia, se puede observar que ciclos antes la corriente eléctrica se encuentra en 100A pico, al suscitarse el evento la corriente sube a 200A pico aproximadamente, dicha corriente no es de gran magnitud en el alimentador y puede no estar dentro de los ajustes de protección para producirse un disparo y por consecuente la apertura del alimentador, ya que el sistema interpretaría esta corriente como un aumento de carga, mas no como una corriente de cortocircuito.

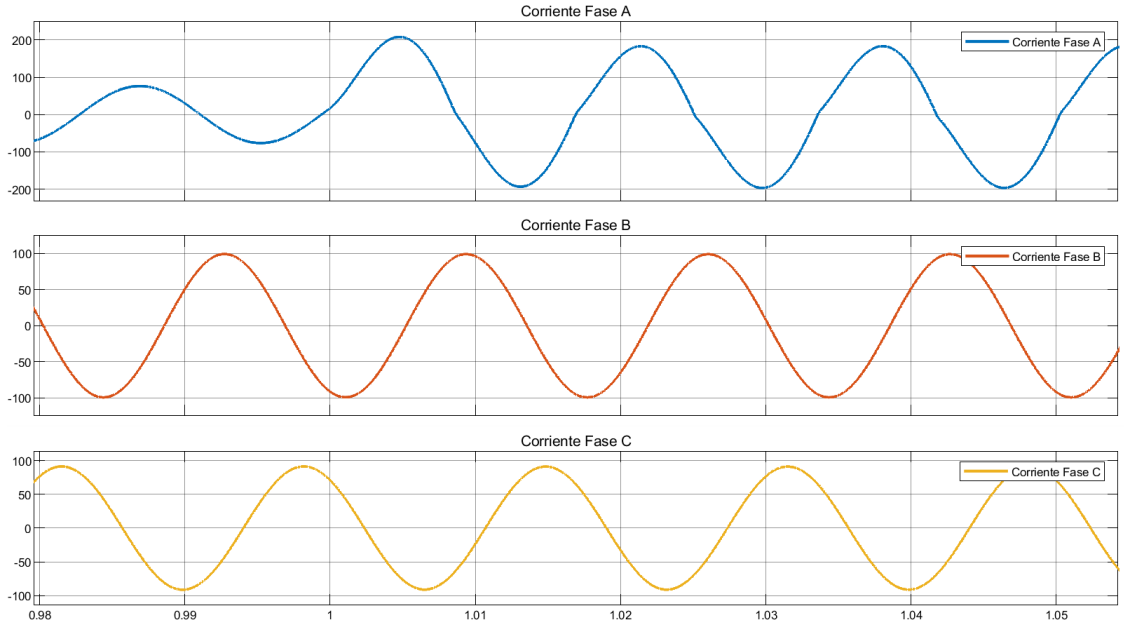


Figura 66. Falla de alta impedancia presentada en la fase A

Al producirse un evento de falla de alta impedancia tal como se puede visualizar en la figura 67, la onda ya no es sinusoidal pura y presenta perturbaciones, además la onda es asimétrica es decir que el pico positivo y negativo no tienen el mismo valor, dichas

características son relevantes, por lo que en este caso se recolecta 500 muestras por ciclo para de esta forma captar la mayor cantidad de datos y por ende estas características que indican que existe un evento de alta impedancia.

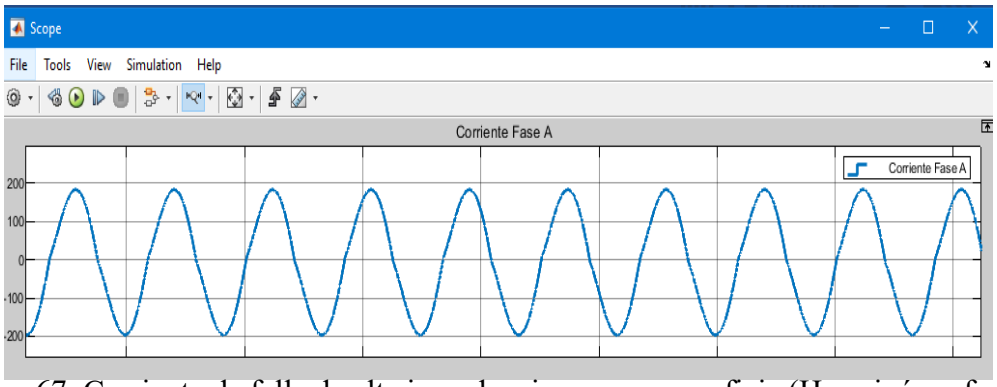


Figura 67. Corriente de falla de alta impedancia en una superficie (Hormigón reforzado)

De la figura 68 se describe que, al producirse un evento de falla, la corriente RMS de la línea incrementa su valor, en este caso en el primer evento de falla se simula un contacto de la línea energizada con una superficie de alta impedancia la cual es el Hormigón reforzado, la corriente nominal se sitúa en los 50A, en el evento de falla incrementa aproximadamente 80A alcanzando un valor de 140A aproximadamente (intervalo de tiempo de 1 a 2 segundos). Durante los 4 a 6 segundos se simula una falla de la línea energizada con pasto seco, esta superficie tiene un aporte de corriente de aproximadamente 30A con lo cual se visualiza que la corriente RMS alcanza un valor de 85A, con esto se denota que al producirse una falla de alta impedancia, la corriente es muy baja para producirse una desconexión en el alimentador de energía, lo cual es muy peligroso, ya que el conductor se encuentra energizado, pudiendo provocar algún accidente, incluso algún incendio, debido al arco eléctrico entre el conductor energizado y la superficie de contacto.

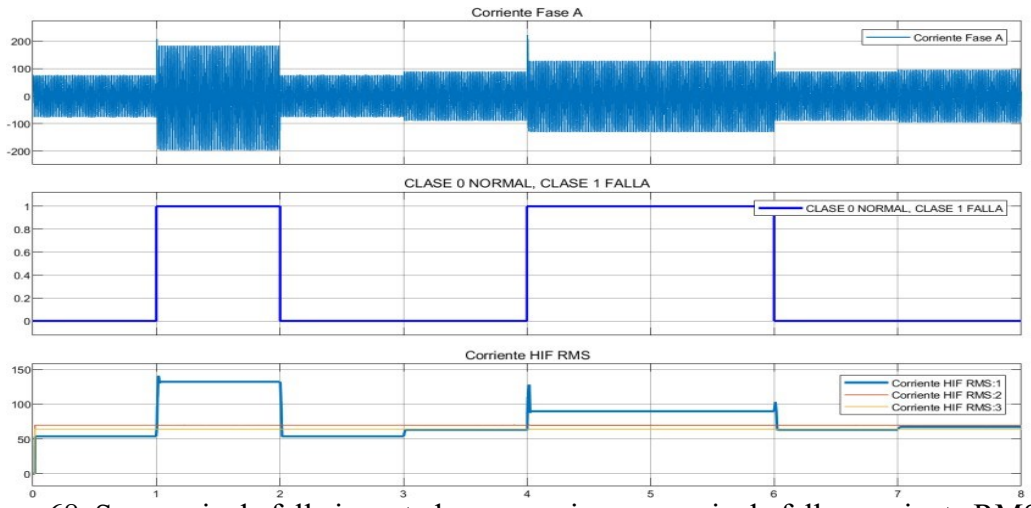


Figura 68. Secuencia de falla inyectada, presencia o ausencia de falla, corriente RMS

Los datos que se muestran en las tablas 13 y 14 muestran las corrientes de fase obtenidas del ambiente simulado ante las condiciones de carga del 80% y 25% de la capacidad nominal del alimentador de distribución eléctrica. En condiciones de carga nominal del 80%, las corrientes RMS en todas las fases son significativamente mayores que en condiciones de carga nominal del 25%, esto es esperado debido a la menor demanda de corriente eléctrica. A medida que la distancia de la falla se incremente, las corrientes de fase también aumentan, esto se puede deber a la mayor impedancia asociada con fallas más lejanas, que incrementa la corriente de cortocircuito requerida para mantener el flujo de energía. En resumen, las fallas de alta impedancia producen un incremento sutil de la corriente eléctrica, lo cual es crítico para la detección y los sistemas de protección eléctrica.

Tabla 13. Datos de simulación obtenidos, condición de carga 80%

Datos de simulación obtenidos									
	Corriente Fase A			Corriente Fase B			Corriente Fase C		
Condición	I pico [A]	ángulo	RMS	I pico [A]	ángulo	RMS	I pico [A]	ángulo	RMS
Carga 80%	95,78	9,94°	67,73	116,2	-115,4°	82,19	109,2	126,6°	77,23
Falla 1km	158,7	-1,24°	112,5	157,8	-121,1°	111,9	175,8	114,9°	124,6
Falla 2km	162,4	-2,23°	115,1	158,4	-121,4°	112,3	177,6	114,4°	125.8
Falla 3km	181,4	-7,40°	128,5	162,3	-123,1°	115	184,6	112,3°	130,7
Falla 4km	184	-8,03°	130,3	190,5	-130,5°	134,9	190,9	110,4°	135,2

Tabla 14. Datos de simulación obtenidos, condición de carga 25%

Datos de simulación obtenidos									
	Corriente Fase A			Corriente Fase B			Corriente Fase C		
Condición	I pico [A]	ángulo	RMS	I pico [A]	ángulo	RMS	I pico [A]	ángulo	RMS
Carga 25%	27,48	17,93°	19,43	37,46	-108,8°	26,49	34,59	134,3°	24,46
Falla 1km	149,7	0,56°	106,2	149,6	-119,4°	106,1	154,8	119,2°	109,8
Falla 2km	151,3	-0,9°	107,3	147,5	-119,7°	104,6	153,4	118,8°	108.8
Falla 3km	155,3	-2,3°	110	147,9	-120,4°	104,9	154,8	118,0°	109,7
Falla 4km	155,5	-2,5°	110,2	158	-124,3°	111,9	158,4	116,3°	112,3

3.5.2 Análisis de métricas obtenidas correspondientes a las arquitecturas de IA desarrolladas

En esta sección se evalúan las métricas de rendimiento tales como la precisión, exactitud, *recall*, F1-score, ROC-AUC, obtenidas al finalizar el entrenamiento de cada uno de los modelos propuestos de inteligencia artificial. Cada una de estas métricas se evalúa sobre el conjunto de validación o prueba, que en esta propuesta representa el 20% del total de los datos utilizados para el entrenamiento. Evaluar el rendimiento en muestras no vistas durante el entrenamiento presenta una ventaja, ya que proporcionará una evaluación más realista de la capacidad de generalización de cada modelo. Además, se realiza un análisis de las matrices de confusión de cada modelo para analizar su capacidad de predicción.

Del primer modelo MPL de siete capas entrenado se obtuvo las métricas visualizadas en la figura 69, donde consiguió una exactitud (*acurracy*) del 78.98%, lo que indica que dicho porcentaje de muestras del conjunto de evaluación fueron clasificadas correctamente, tanto para la clase 0 (ausencia de falla) como clase 1 (falla de alta impedancia). Una precisión obtenida del 72.25% demuestra que el modelo cada vez que se presente un evento de clase 1 el 72.25% de veces será correcta. Un *recall* de 0.9145 conseguido indica que el 91.45% de todas las fallas de alta impedancia (clase 1) que se encuentren presentes en el conjunto de datos serán detectadas correctamente, mientras que el 8.55% de las fallas no serán detectadas, estas se consideran como falsos negativos. El f1-score obtenido de 0.80 sugiere que el modelo puede manejar adecuadamente la identificación de fallas de alta impedancia como la reducción de falsas alarmas, mientras que un valor de ROC-AUC de 0.9396 demuestra que la red MPL 1 puede ser competente para distinguir entre una forma de onda de corriente nominal y una onda de falla de alta impedancia.

La matriz de confusión de la primera arquitectura entrenada demuestra el siguiente análisis:

- 509 ventanas de datos son verdaderos negativos (TN): Predicciones correctas como clase 0 (ausencia de falla)
- 0 ventanas de datos son falsos positivos (FP): Del conjunto de datos de evaluación no se predijeron incorrectamente eventos de clase 1 (detección de falla de alta impedancia)

- 1 ventana de datos son falsos negativos (FN): Incorrectamente predichos como clase 0 (ausencia de falla)
- 489 ventanas de datos son verdaderos positivos (TP): Correctamente predichos como clase 1 (presencia de fallad de alta impedancia)

```
32/32 [==============================] - 1s 8ms/step
Accuracy: 0.998998998998999
Precision: 1.0
Recall: 0.9979591836734694
F1-score: 0.9989785495403473
ROC-AUC Score: 0.9991219277494888
Confusion Matrix:
[[509   0]
 [  1 489]]
```

Figura 69. Métricas de evaluación obtenidas al entrenar la red MPL 1

La figura 70 muestra las curvas de pérdidas y precisión de entrenamiento y validación obtenidas, en donde se observa que la precisión de entrenamiento se sitúa en un valor promedio de 0.97 lo cual indica la capacidad del modelo para clasificar correctamente los datos de entrenamiento. Sin embargo, en la precisión de validación se visualizan fluctuaciones irregulares, lo que es producto de haber utilizado un batch size muy pequeño durante el entrenamiento produciendo una mayor variabilidad en la actualización de los pesos del modelo.

Para el caso de la curva de la pérdida de entrenamiento y validación se puede interpretar que el modelo se encuentra bien ajustado y realizando un aprendizaje óptimo, ya que ambas pérdidas se convergen en un valor cercano a 0 cerca alrededor de la época 20. Esto sugiere que el modelo puede generalizar de manera adecuada ante nuevos datos. Además, se visualiza que se obtiene un modelo sin presencia de sobreajuste ni subajuste.

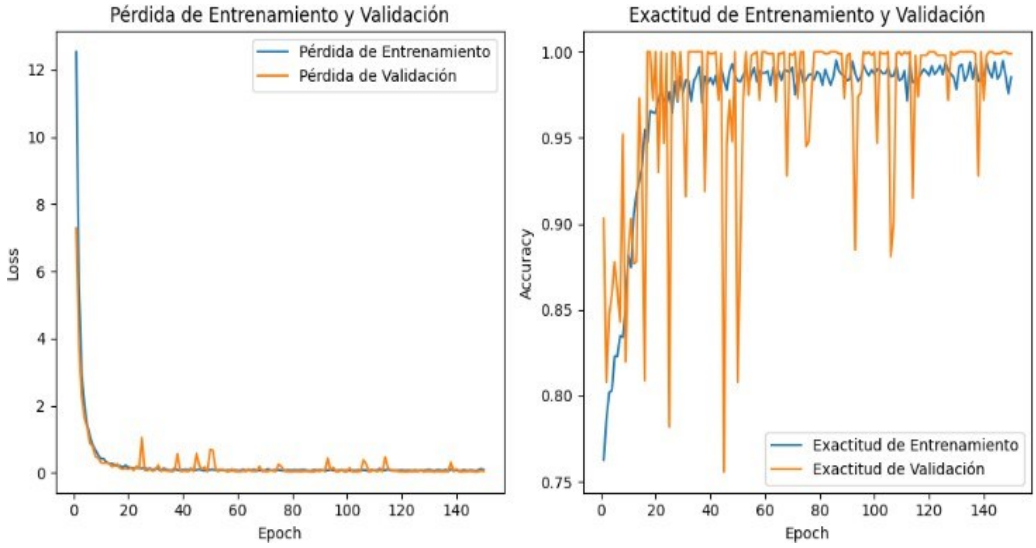


Figura 70. Pérdidas y precisión de entrenamiento y evaluación red MPL 1

En la figura 71 se visualiza las métricas de evaluación de la segunda arquitectura MPL de nueve capas totalmente conectadas, donde se obtuvieron resultados altamente eficientes. A continuación, se analiza en detalle los resultados obtenidos:

- Exactitud (acurracy): Del conjunto de datos destinado para la evaluación, el modelo predijo correctamente el 100% de las muestras. Esto incluye tanto la clase 0 indicando que no existe ninguna falla, como la clase 1 indicando la detección de una falla de alta impedancia.
- Precisión: De igual forma una precisión del 100% obtenida, indica que el modelo es muy eficaz en la predicción de eventos de clase 1, es decir que las predicciones positivas son correctas en su totalidad. Un Recall del 100% asegura que el modelo fue capaz de identificar todas las muestras de clase 1.
- El valor F1-score de 1, confirma un equilibrio entre la métrica precisión y el recall, lo que resalta la robustez del modelo.
- Para el caso del valor de 1 en la métrica ROC-AUC indica que este modelo tiene una capacidad perfecta para diferenciar una forma de onda de corriente nominal de una onda de corriente eléctrica con las perturbaciones que se dan cuando se presenta un evento de alta impedancia en las líneas eléctricas de distribución.

Además, la matriz de confusión obtenida refleja el excelente desempeño del modelo sobre el conjunto de datos de evaluación, en esta se obtuvo que 509 ventanas de datos resultaron verdaderos negativos es decir se predijeron correctamente como clase 0, 490 ventanas de datos son verdaderos positivos es decir el modelo tuvo la capacidad de predecir correctamente todas las muestras de clase 1, mientras que se obtuvo 0 casos de falsos positivos como falsos negativos es decir ventanas de datos que se clasificaron incorrectamente. El análisis de la matriz de confusión respalda las métricas eficientes obtenidas, demostrando que el modelo no cometió ningún error en la clasificación de las muestras. Esto nos da un preámbulo de una alta fiabilidad de este modelo en la detección de fallas de alta impedancia.

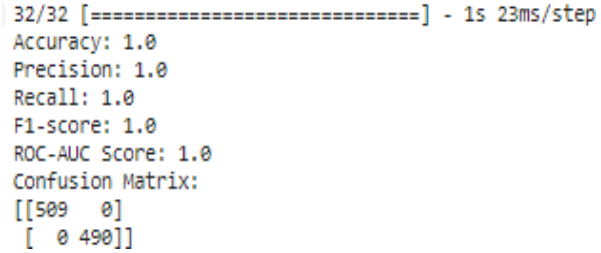


Figura 71. Métricas de evaluación obtenidas al entrenar la red MPL 2

De la siguiente figura 72, se muestra que el modelo cerca de la época 40 muestra una convergencia a un valor inferior a 0.2, lo que indica un correcto aprendizaje de las características de los datos de entrenamiento minimizando el error en las predicciones de las etiquetas de salida, a diferencia de la curva de pérdidas y precisión de entrenamiento y validación de la primera arquitectura MPL, está presenta un pequeño incremento de las pérdidas de 0.25 a 0.50 durante aproximadamente 20 épocas lo que puede inferirse como un ajuste de parámetros debido a que el modelo aún no ha encontrado los patrones adecuados.

Conjuntamente se muestra la curva de la exactitud del modelo, en donde se observa que la exactitud de entrenamiento y validación alcanzan un valor muy cercano a 1 cerca de la época 40 y se mantiene cercano a este valor durante 110 épocas restantes, lo que indica que el modelo está bien entrenado y ha encontrado los pesos óptimos de un aprendizaje eficaz que le permiten predecir correctamente tanto las muestras del conjunto de entrenamiento como el de validación. Que ambos valores de entrenamiento y validación de la curva de exactitud sean muy similares y cercano a 1 indican que el modelo no se encuentra sobre ajustado, caso contrario se visualizaría valores altos de la curva de exactitud de entrenamiento, pero la de exactitud de validación tendría valores diferentes.

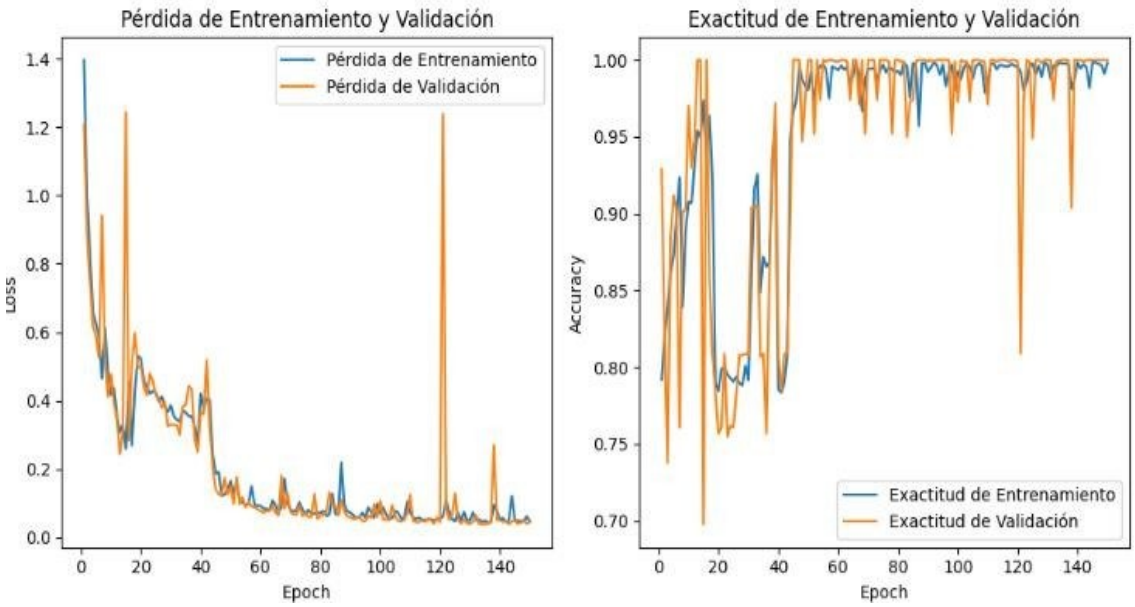


Figura 72. Pérdidas y precisión de entrenamiento y evaluación red MPL 2

Las métricas de evaluación obtenidas al entrenar el modelo de la primera arquitectura propuesta de red neuronal convolucional, muestra de manera general un buen rendimiento de predicción de las clases negativas y positivas, presentes en el conjunto de evaluación, ya que las métricas presentadas se sitúan en valores superiores a 0.99 (Ver figura 73). De la matriz de confusión de esta red neuronal convolucional se muestra el siguiente análisis:

- No se obtuvo falsos negativos, es decir el modelo no clasifico erróneamente eventos de clase 0.
- Verdaderos negativos: 506 ventanas de datos del conjunto de datos de validación se predijeron correctamente como clase 0.
- Verdaderos Positivos: 490 ventanas de datos fueron predichas correctamente como clase 1.
- Falsos positivos: 3 ventanas de datos se predijeron como clase 1, cuando en realidad eran clase 0.

```
32/32 [==============================] - 11s 323ms/step
Accuracy: 0.996996996996997
Precision: 0.9939148073022313
Recall: 1.0
F1-score: 0.9969481180061037
ROC-AUC Score: 1.0
Confusion Matrix:
[[506   3]
 [  0 490]]
```

Figura 73. Métricas de evaluación obtenidas al entrenar la red Convolucional 1

La figura 74 muestra el proceso de aprendizaje de la red neuronal convolucional de siete capas propuesta, donde se observa que las pérdidas de precisión y validación convergen cerca de la época 30 a un valor muy cercano a 0, indicando que el modelo aprendió los patrones de los datos de entrenamiento y encontró los pesos adecuados para predecir y generalizas las respectivas clases de los datos de entrenamiento y validación. La curva de precisión obtenida en este modelo demuestra que este puede alcanzar un buen desempeño al detectar eventos de clase positiva o clase 1. En comparación con las gráficas obtenidas de las pérdidas y precisión del entrenamiento realizado en las dos arquitecturas MPL propuestas, esta muestra un entrenamiento suave sin alta variabilidad en los pesos del

modelo debido a que se seleccionó un *batch size* de 32 para gestionar de manera adecuada los recursos computacionales y una rápida convergencia del respectivo modelo.

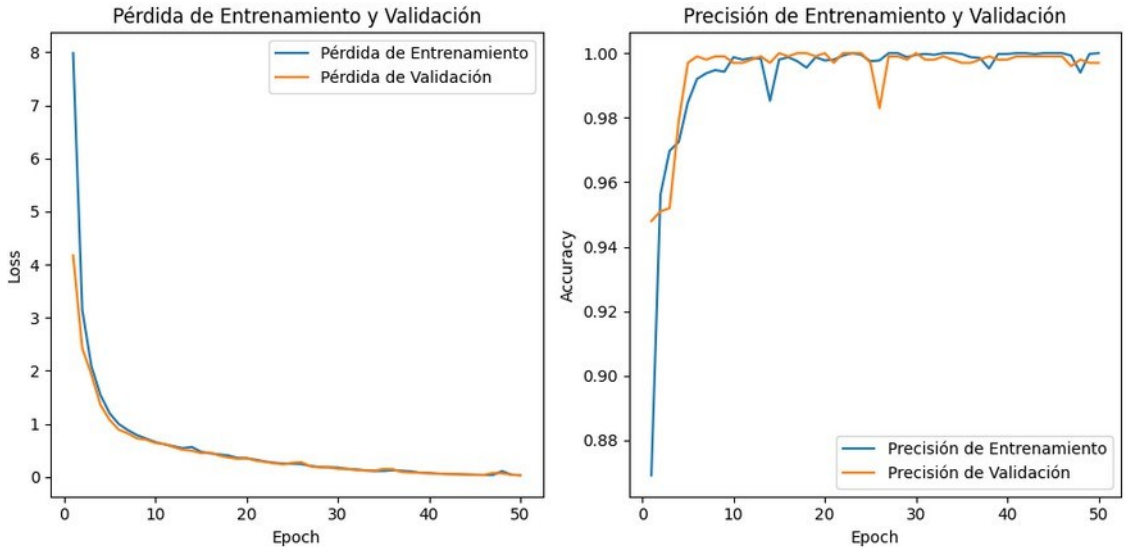


Figura 74. Pérdidas y precisión de entrenamiento y validación de la red Convolucional 1

Las métricas obtenidas del entrenamiento de la red convolucional de doce capas demuestran un desempeño y aprendizaje eficaz. En este caso, las métricas son idénticas a las de la segunda arquitectura MPL entrenada, donde también se lograron resultados eficientes (Ver figura 75). Esto indica que el modelo entrenado fue capaz de clasificar correctamente las clases presentes en el conjunto de validación. Además, la consistencia entre las dos arquitecturas sugiere una robustez en el diseño del modelo y una capacidad de generalización adecuada frente a diferentes conjuntos de datos. Estas métricas de evaluación obtenidas refuerzan la viabilidad de utilizar redes convolucionales profundas para tareas complejas de clasificación, como en esta propuesta, donde es crucial detectar minuciosas características de asimetría y no linealidades presentes en una forma de onda sinusoidal de corriente eléctrica.

```
32/32 [==============================] - 7s 219ms/step
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1-score: 1.0
ROC-AUC Score: 1.0
Confusion Matrix:
[[509   0]
 [  0 490]]
```

Figura 75. Métricas de evaluación obtenidas al entrenar la red convolucional 2

A continuación, se muestra la gráfica 76 que detalla el proceso de entrenamiento de la red neuronal convolucional a lo largo de 50 épocas, destacando el progreso de las pérdidas y exactitud tanto para el conjunto de entrenamiento como para el de validación. De dichas gráficas se detalla el respectivo análisis:

- El modelo empieza con un desempeño considerablemente bueno en las pérdidas de validación (0.090), aunque con una alta perdida en el entrenamiento (2.6), mientras que para el caso de la precisión de entrenamiento y validación un 0.83 y 0.95 respectivamente.
- En la segunda época el modelo reduce drásticamente las pérdidas de entrenamiento y validación, y alcanza una exactitud del 100%, lo que sugiere que el modelo aprendió de manera eficiente los datos de entrenamiento.
- Debido a que las pérdidas y precisión son perfectas en el conjunto de validación, podría ser una señal de sobre ajuste si los datos de validación no son suficientemente representativos de datos no vistos.

En resumen, los resultados indican que la red neuronal convolucional ha aprendido a clasificar con una precisión extremadamente y pedidas muy bajas lo que es ideal para esta aplicación donde la precisión es crítica. Pese a estos resultados óptimos obtenidos de esta red neuronal convolucional, es de vital importancia, evaluar el modelo con datos no vistos durante el entrenamiento para asegurar su capacidad de generalización.

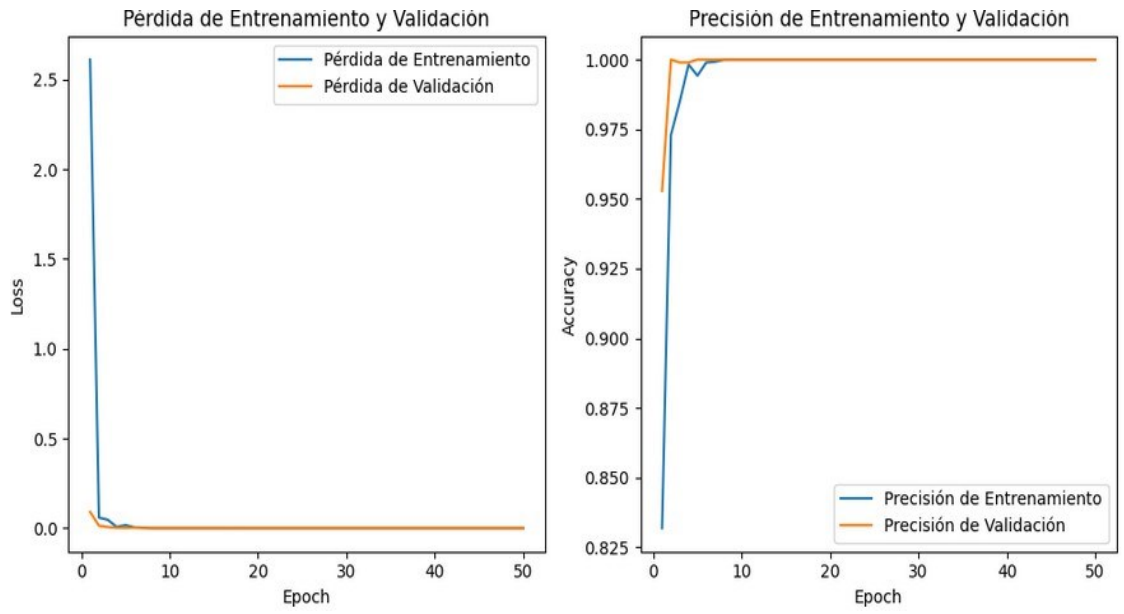


Figura 76. Pérdidas y precisión de entrenamiento y validación de la red convolucional 2

Al entrenar el quinto modelo de inteligencia artificial SVM propuesto para la detección de fallas de alta impedancia, se obtuvo una precisión del 99.59% es decir el modelo clasifico correctamente el total de las muestras del conjunto de validación. De la matriz de confusión obtenida, la cual se muestra en la figura 77, describe un desempeño óptimo en la clasificación de las clases 0 y 1 presentes en los datos de evaluación, 509 ventanas de datos se clasificaron como verdaderos negativos, es decir se clasificaron correctamente como clase 0, 486 ventanas de datos se clasificaron como verdaderos positivos, en este caso el modelo predijo correctamente los eventos de falla de alta impedancia presentes en el conjunto de datos de evaluación. Como falsos negativos se obtuvo 4 ventanas de datos en donde el modelo predijo como clase 0, cuando en realidad se presentaba ventanas de datos con eventos de falla de alta impedancia, mientras que no se obtuvo predicciones de falsos positivos, con esto se resalta la alta precisión conseguida al evaluar y clasificar las ventanas de datos del conjunto de evaluación.

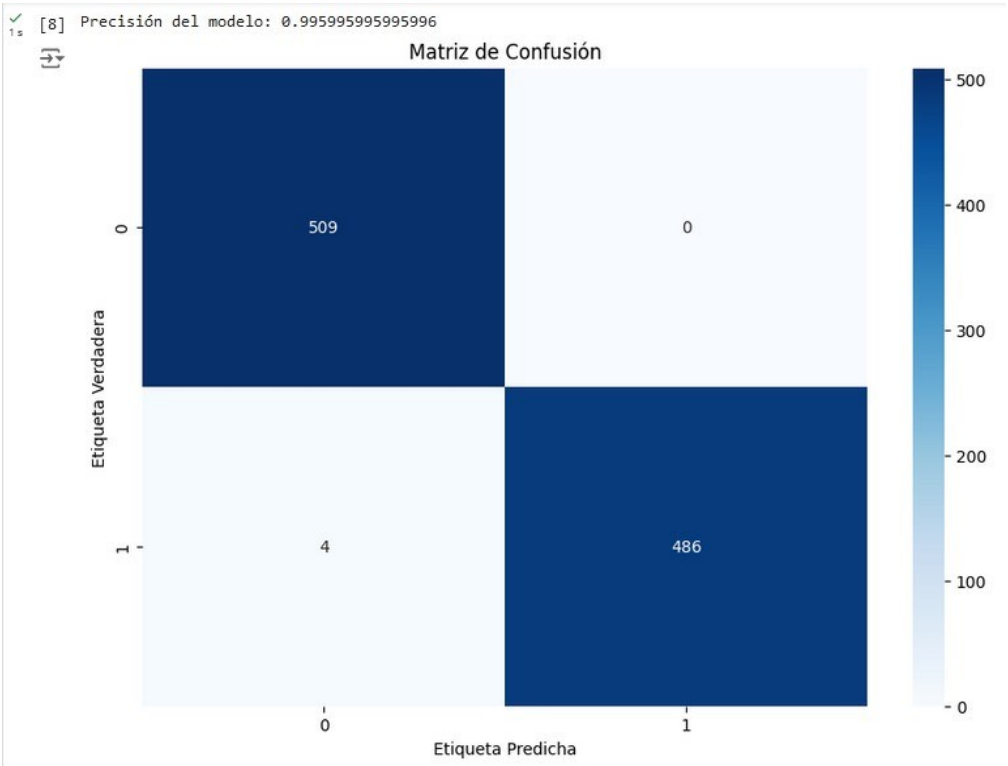


Figura 77. Precisión y matriz de confusión obtenida del entrenamiento del algoritmo SVM

3.5.3 Evaluación de los modelos de IA entrenados ante nuevos conjuntos de datos

Generando nuevos conjuntos de datos con presencia y ausencia de falla en el entorno de simulación MATLAB/Simulink, diferentes a los datos ingresados en el entrenamiento de los modelos, se evalúan el rendimiento de cada uno de los modelos de inteligencia artificial entrenados, para de esta forma seleccionar el modelo que presenta las métricas de evaluación eficientes y la mínima tasa de error en la detección de fallas de alta impedancia. El modelo seleccionado será puesto en marcha en un sistema embebido tal como lo es una tarjeta Raspberry Pi 4.

Para evaluar la capacidad de predicción de los modelos de inteligencia artificial entrenados, se propone tres esquemas de conjuntos de datos que contienen diferentes escenarios de operatividad (Ver Tabla 15, 16 y 17), el primero durante 420 ciclos, lo cual da un periodo aproximado de 7 segundos, mientras que el segundo y tercer conjunto de datos contiene un periodo extenso de aproximadamente 16.6 segundos. Durante estos nuevos datos se inserta condiciones que se pueden suscitar en el ámbito real en las redes eléctricas de distribución, tal como un aumento de carga en un instante de tiempo, además de las diferentes superficies de contacto a la que se puede producir una falla de alta impedancia, las cuales fueron mostradas en las tablas 1 y 2.

Tabla 15. Primer esquema de prueba para validación de los modelos entrenados

Superficie	N/A	Concreto Reforzado		N/A		Hierba mojada		N/A		Césped mojado	
Condición/ Clase	Condición Normal Clase 0	Inserción Falla Clase 1		Condición Normal Clase 0		Inserción Falla Clase 1		Condición Normal Clase 0		Inserción Falla Clase 1	
				Aumento de Carga				Aumento de Carga			
tiempo [s]	0s	0.33s	1s	1.5s	2s		3s	3.5s	4s		5s

Tabla 16. Segundo esquema de prueba para validación de los modelos entrenados, 25% de carga eléctrica

Superficie	N/A		Pasto seco		N/A			Pasto seco		N/A			Pasto seco	
Condición/ Clase	Condición Normal Clase 0		Inserción Falla Clase 1		Condición Normal Clase 0			Inserción Falla Clase 1		Condición Normal Clase 0			Inserción Falla Clase 1	
					Aumento de Carga					Aumento de Carga				
tiempo [s]	0s		0.035s	2s	3s	4s		6s		6.5s	8s		10s	

Tabla 17.Tercer esquema de prueba para validación de los modelos entrenados

Superficie	N/A		Rama de árbol		N/A		Rama de árbol		N/A	
Condición/ Clase	Condición Normal Clase 0		Inserción Falla Clase 1		Condición Normal Clase 0		Inserción Falla Clase 1		Condición Normal Clase 0	
					Aumento de Carga				Aumento de Carga	
tiempo [s]	0s	2s		4s		6s		8s		10s

En la figura 78 y 79 se muestra los respectivos conjuntos de datos generados en el entorno de simulación MATLAB/Simulink para evaluar las diferentes arquitecturas de redes neuronales entrenadas, en ellas se visualiza la forma de onda de operación normal de corriente eléctrica, la cual es una onda sinusoidal pura, mientras que la onda de la corriente eléctrica que se da en una falla de alta impedancia presenta perturbaciones, tales como la asimetría en la que los valores picos máximo positivo y negativo presentan diferentes amplitudes, además de no linealidades presentes en esta onda . Además, en estas figuras se muestra una señal cuadrada la cual hace referencia a la clase de salida respectiva la cual únicamente presenta dos instancias, clase 0 indicando que en el alimentador existe un flujo de corriente eléctrica en condiciones normales, mientras que la clase 1 indica que existe una falla de alta impedancia en la línea eléctrica de distribución, la cual puede ser causada por una de las superficies de baja conductividad eléctrica mostrada en la Tabla 2.

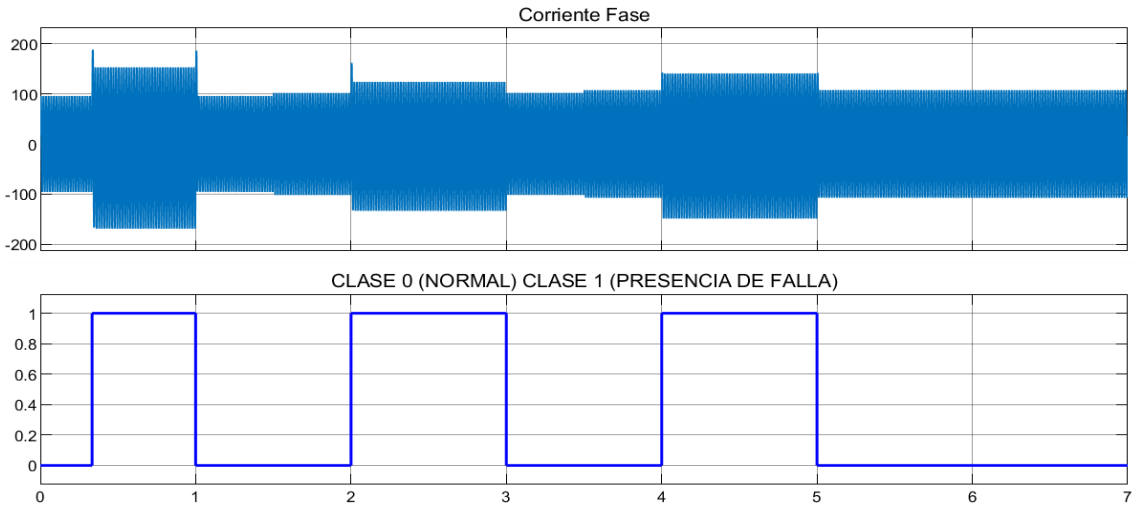


Figura 78. Señal de corriente de normal operación y de falla del conjunto del primer conjunto de evaluación

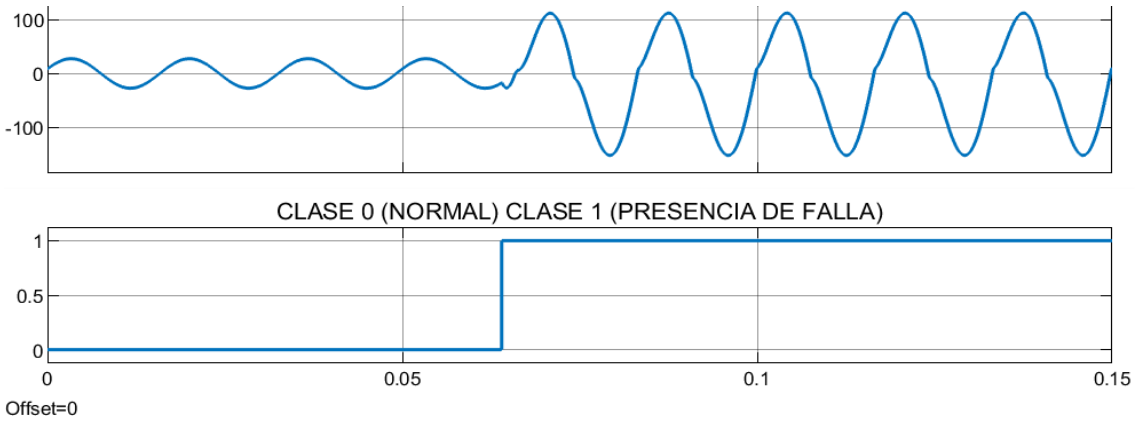


Figura 79. Onda de corriente eléctrica de normal operación y de falla de alta impedancia

Las siguientes figuras 80, 81, 82, 83, 84 y 85 muestran el desempeño de las dos arquitecturas *MultiPerceptron Layer* de siete y nueve capas respectivamente mediante diagramas circulares en las que se detalla el porcentaje de detección obtenido para las dos clases 0 y 1. Un porcentaje de predicción del 37.5% se obtuvo en las muestras de clase 0 mientras que en las muestras de clase 1 se obtuvo un porcentaje superior al 87.53%, resaltando que este modelo tiene la capacidad para predecir conjuntos de datos de fallas de alta impedancia, mientras que los datos de operación nominal los predice incorrectamente. En el caso de la red MPL de nueve capas se obtuvo un 99.8% de predicciones correctas de datos de falla de alta impedancia, mientras que un 74.86% de predicciones correctas para datos de clase 0, lo que demuestra que a medida que se agrega capas y se hace más profunda la arquitectura MPL, puede reducirse el margen de error en las predicciones y generalizar de manera eficiente ante nuevos datos

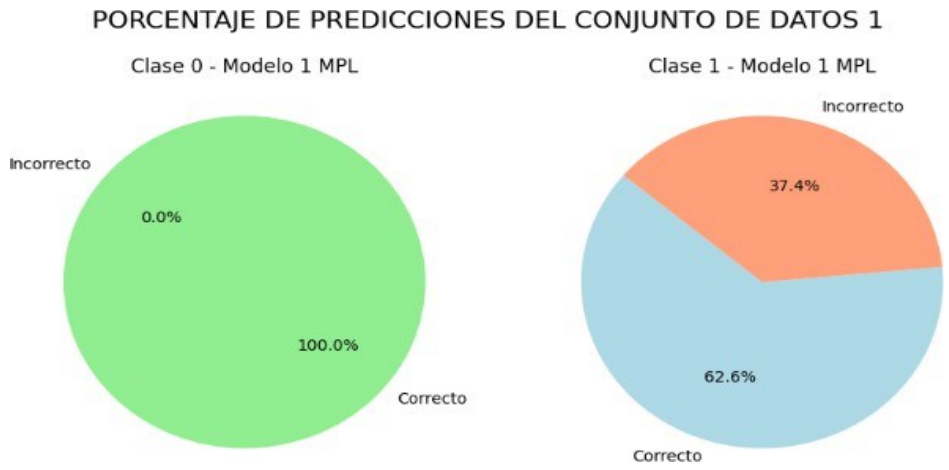


Figura 80.Porcentaje de predicciones de la primera red neuronal MPL (Secuencia de datos 1)

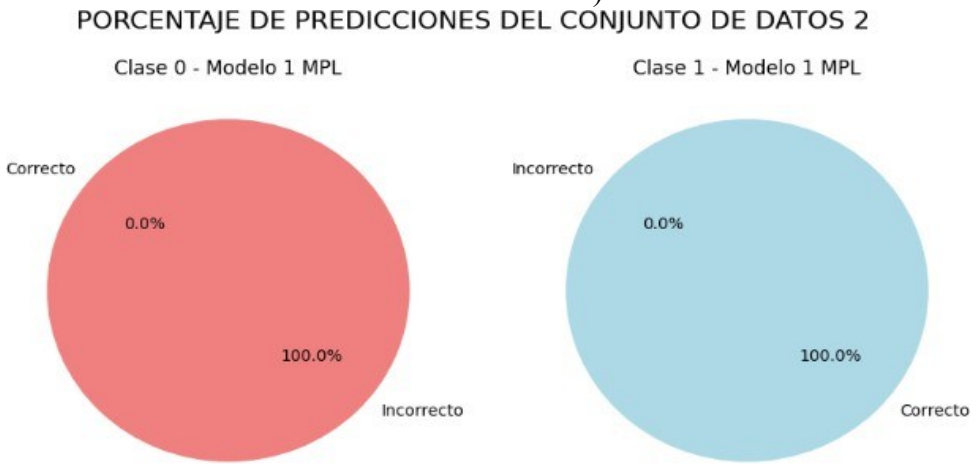


Figura 81. Porcentaje de predicciones de la primera red neuronal MPL (Secuencia de datos 2)

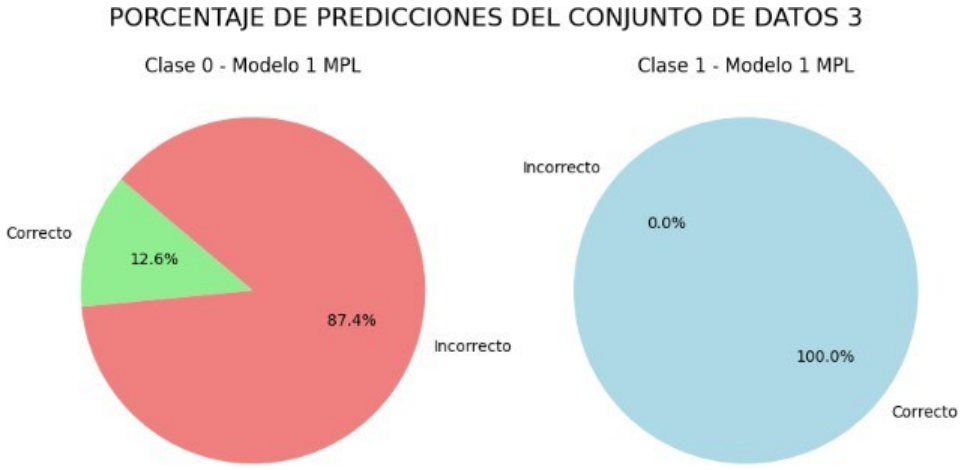


Figura 82. Porcentaje de predicciones de la primera red neuronal MPL (Secuencia de datos 3)

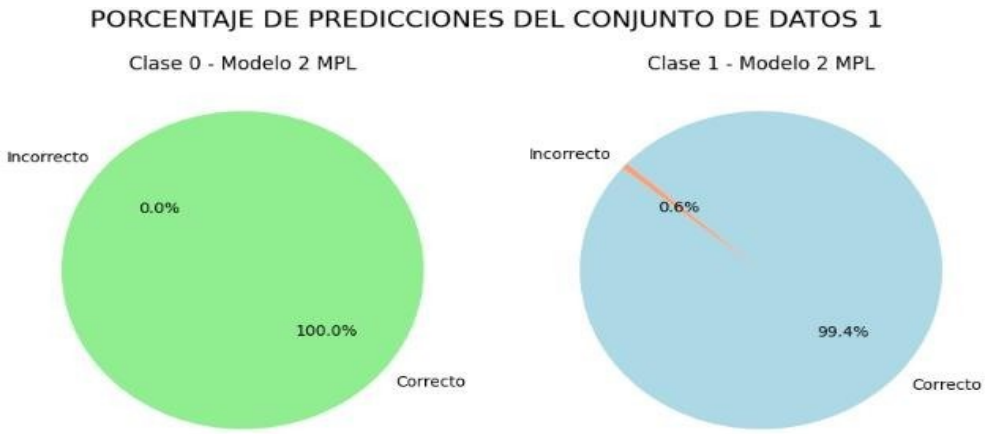


Figura 83. Porcentaje de predicciones de la segunda red neuronal MPL (Secuencia de datos 1)

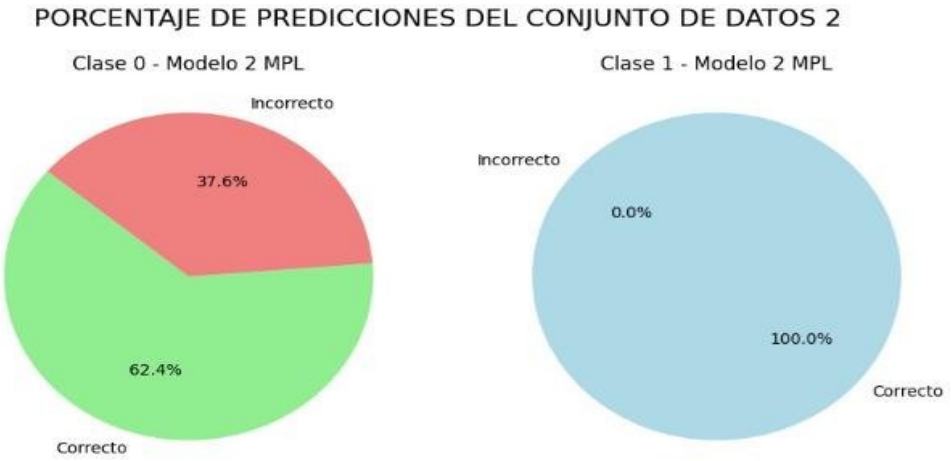


Figura 84. Porcentaje de predicciones de la segunda red neuronal MPL (Secuencia de datos 2)

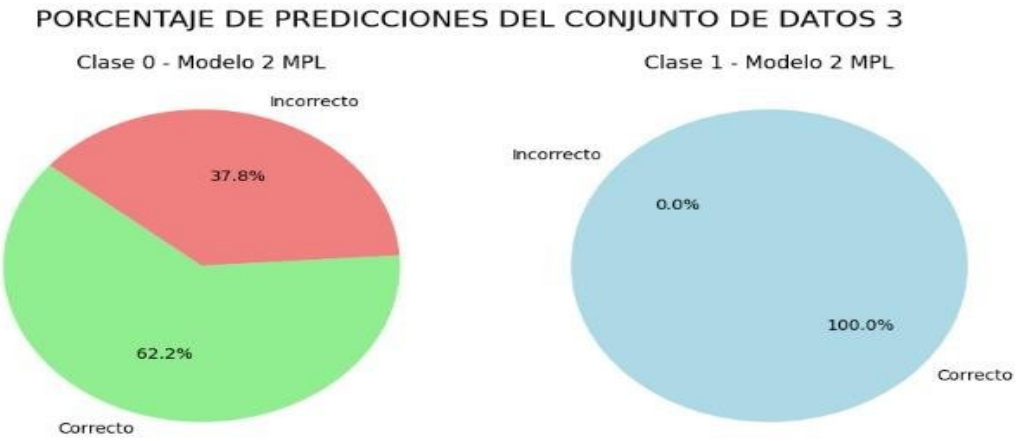


Figura 85. Porcentaje de predicciones de la segunda red neuronal MPL (Secuencia de datos 3)

La figura 86 muestra los resultados obtenidos al evaluar la red neuronal convolucional de siete capas con tres nuevos conjuntos de datos no vistos por el modelo durante el entrenamiento. Para la detección correcta de eventos de operación normal (clase 0), se

logró un porcentaje de predicción promedio del 17.93%, mientras que un 82.06% de las predicciones fueron incorrectas de porcentaje, es decir, estas muestras se clasificaron como fallas cuando en realidad eran muestras de normal operación. Por otro lado, las figuras 87 y 88 muestran las predicciones realizadas por la red convolucional 1, en la detección correcta de eventos de fallas de alta impedancia (clase 1) se obtuvo un porcentaje del 72.76%, frente a un 27.23% de predicciones incorrectas, lo que significa que el modelo clasificó estos eventos como clase 0 cuando en realidad eran fallas de alta impedancia.

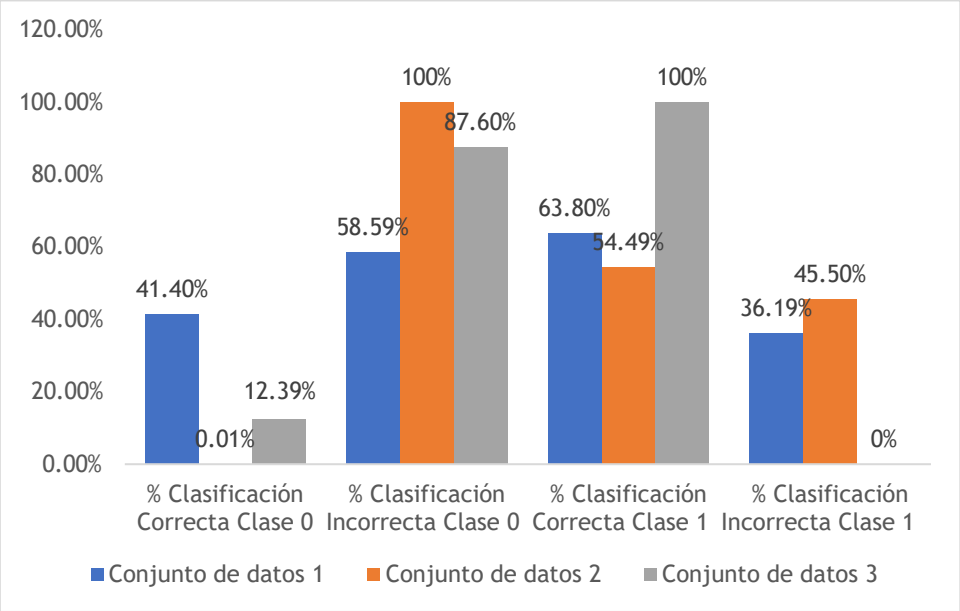


Figura 86. Porcentaje de predicciones de la tercera arquitectura red neuronal convolucional

```
Predicciones vs. Salida Real:
Muestra 1 - Predicha: 1, Real: 0
Muestra 2 - Predicha: 1, Real: 0
Muestra 3 - Predicha: 1, Real: 0
Muestra 4 - Predicha: 1, Real: 0
Muestra 5 - Predicha: 1, Real: 0
Muestra 6 - Predicha: 1, Real: 0
Muestra 7 - Predicha: 1, Real: 0
Muestra 8 - Predicha: 1, Real: 0
Muestra 9 - Predicha: 1, Real: 0
Muestra 10 - Predicha: 1, Real: 0
Muestra 11 - Predicha: 1, Real: 0
Muestra 12 - Predicha: 1, Real: 0
Muestra 13 - Predicha: 1, Real: 0
Muestra 14 - Predicha: 1, Real: 0
Muestra 15 - Predicha: 1, Real: 0
Muestra 16 - Predicha: 1, Real: 0
Muestra 17 - Predicha: 1, Real: 0
Muestra 18 - Predicha: 1, Real: 0
Muestra 19 - Predicha: 1, Real: 0
Muestra 20 - Predicha: 1, Real: 0
```

```
Muestra 120 - Predicha: 1, Real: 1
Muestra 121 - Predicha: 0, Real: 1
Muestra 122 - Predicha: 0, Real: 1
Muestra 123 - Predicha: 0, Real: 1
Muestra 124 - Predicha: 0, Real: 1
Muestra 125 - Predicha: 0, Real: 1
Muestra 126 - Predicha: 0, Real: 1
Muestra 127 - Predicha: 0, Real: 1
Muestra 128 - Predicha: 0, Real: 1
Muestra 129 - Predicha: 0, Real: 1
Muestra 130 - Predicha: 0, Real: 1
Muestra 131 - Predicha: 0, Real: 1
Muestra 132 - Predicha: 0, Real: 1
Muestra 133 - Predicha: 0, Real: 1
Muestra 134 - Predicha: 0, Real: 1
Muestra 135 - Predicha: 0, Real: 1
Muestra 136 - Predicha: 0, Real: 1
Muestra 137 - Predicha: 0, Real: 1
Muestra 138 - Predicha: 0, Real: 1
Muestra 139 - Predicha: 0, Real: 1
Muestra 140 - Predicha: 0, Real: 1
```

Figura 87. Predicciones incorrectas de la red neuronal convolucional 1

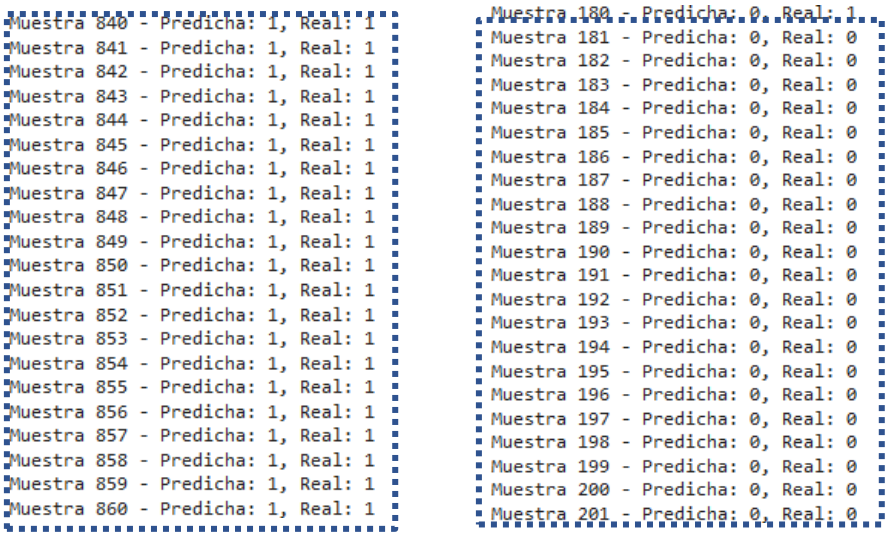


Figura 88. Predicciones correctas de la red neuronal convolucional 1

Se puede visualizar en la siguiente figura, el rendimiento de la red neuronal convolucional propuesta, la cual contiene 12 capas. Esta red neuronal muestra resultados más favorables en comparación con la anterior red convolucional, de manera general se puede interpretar que esta red tiene menor porcentaje de error. Para las muestras de clase 0 esta red clasificó correctamente el 47.76%, mientras que un 52.36% las clasificó de manera incorrecta, es decir, la red neuronal las interpreto como eventos de falla. Los eventos de clase 1 se clasificaron de manera correcta en un 72.86%, mientras que un 27.16% se clasifico incorrectamente como clase 0, cuando en realidad eran eventos de falla de alta impedancia (Ver figura 90). En resumen, en las gráficas 89 y 91, se observa que este modelo presenta mejoras en la detección de eventos de clase 0 en comparativa de la red convolucional de siete capas, mientras que para eventos de clase 1 se mantiene una precisión superior al 70%.

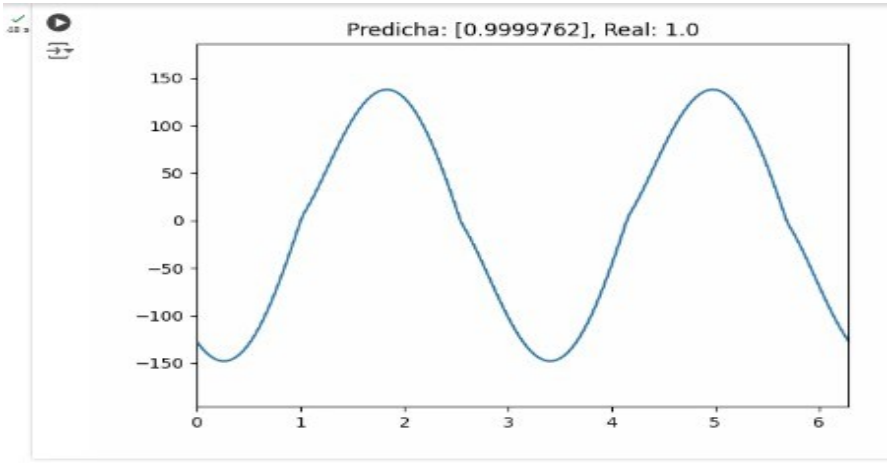


Figura 89. Clasificación correcta de la red neuronal convolucional

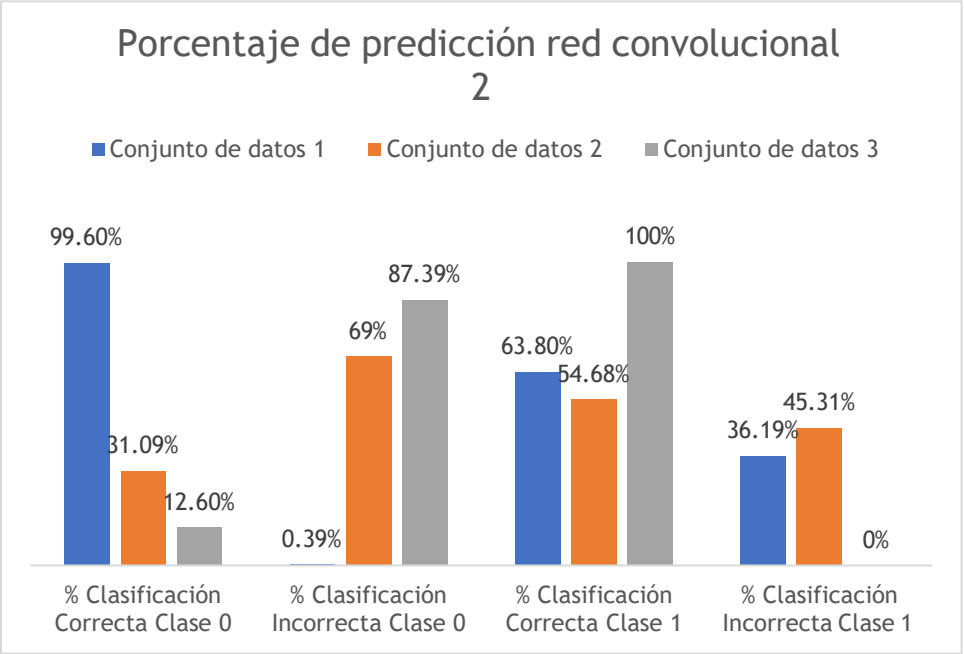


Figura 90. Porcentaje de predicciones de la cuarta arquitectura red neuronal convolucional

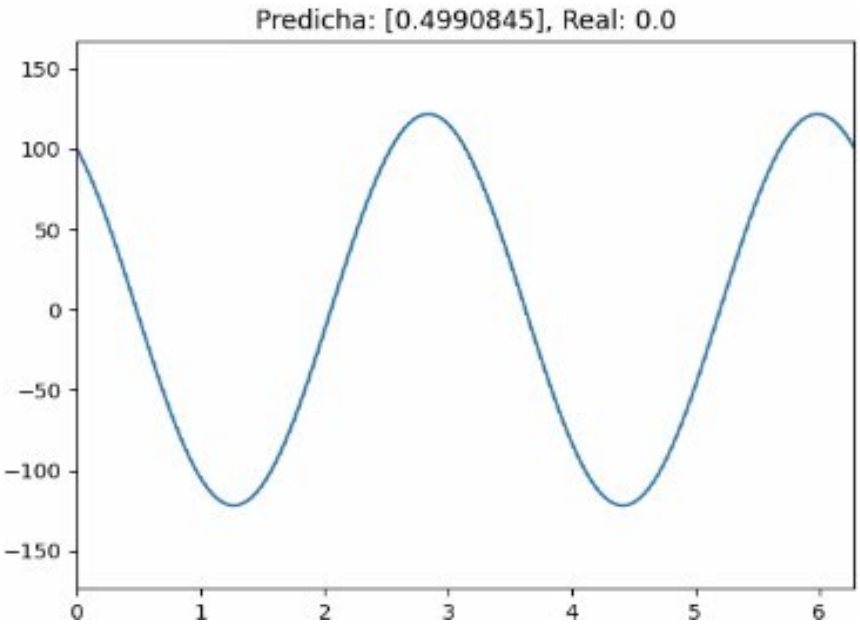


Figura 91. Clasificación correcta de la red neuronal convolucional 2 Clase 0

En el diagrama de barras mostrado en la figura 92, se visualiza el porcentaje de predicción obtenido de la máquina de soporte vectorial, un modelo de aprendizaje automático empleado específicamente para tareas de clasificación y regresión. En el gráfico se visualizan porcentajes muy cercanos al 100% tanto para la clase 0 (ausencia de falla) como para la clase 1 (presencia de falla de alta impedancia). Además, se muestran

porcentajes inferiores de clasificación incorrecta para las respectivas clases. Un 99.78% y un 98.14% de clasificación correcta de las clases 0 y 1 respectivamente, destacan la alta precisión del modelo para distinguir entre estas clases. Por otro lado, solo el 0.21% y 1.81% de las muestras fueron clasificadas incorrectamente. Adicionalmente, la baja tasa de error sugiere una excelente capacidad de generalización del modelo ante nuevos datos, lo cual es vital para su implementación en escenarios reales.

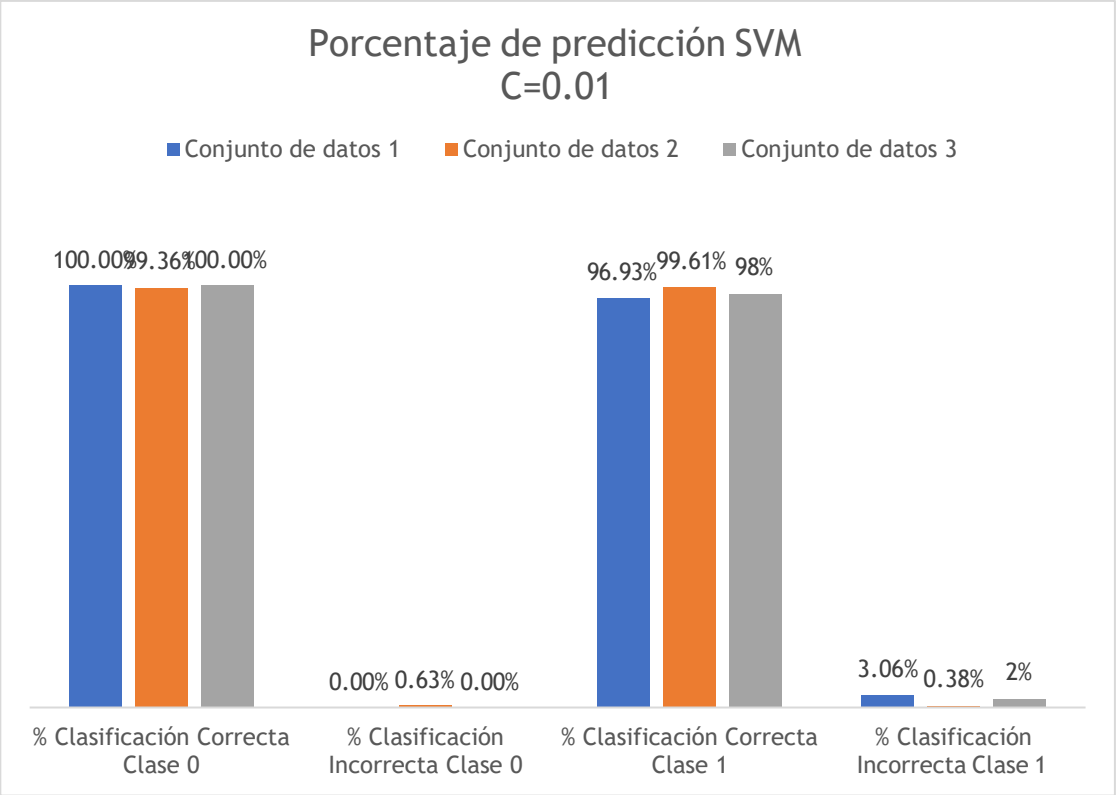


Figura 92. Porcentaje de predicciones de máquina de soporte vectorial (SVM)

A continuación, se muestran las figuras 93 y 94, donde se muestra en primera instancia todas las salidas predichas por la máquina de soporte vectorial, en donde se visualiza las clases del segundo conjunto de datos propuesto en la tabla A, en esta se puede observar que el algoritmo puede predecir correctamente las clases 0 y 1 en su mayoría, a excepción de ciertas muestras en las que el algoritmo de inteligencia artificial predice la clase de manera errónea. En la figura 94 se puede observar en la ventana de dato 359 se tiene una predicción de clase 0, mientras que en la siguiente ventana de dato 360 se tiene una falla de alta de alta impedancia, esto demuestra la alta capacidad del modelo de aprendizaje automático para discernir eficientemente entre un evento de operación nominal y una falla de alta impedancia.

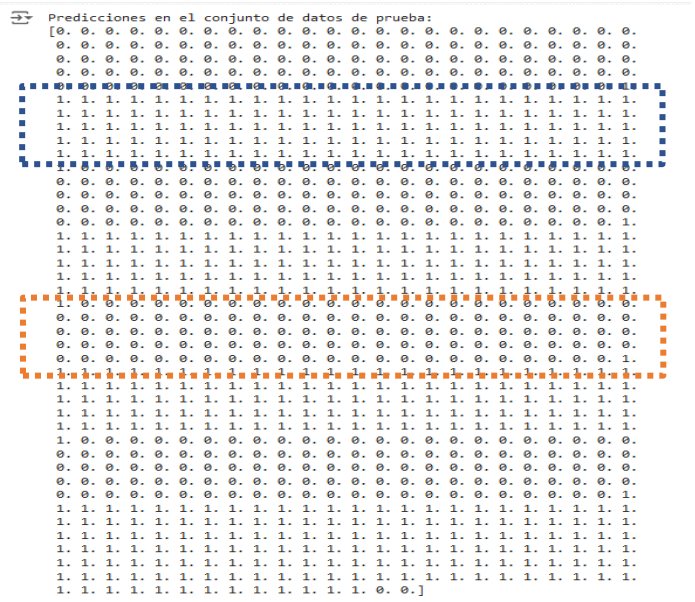


Figura 93. Salida predicha por parte de la SVM

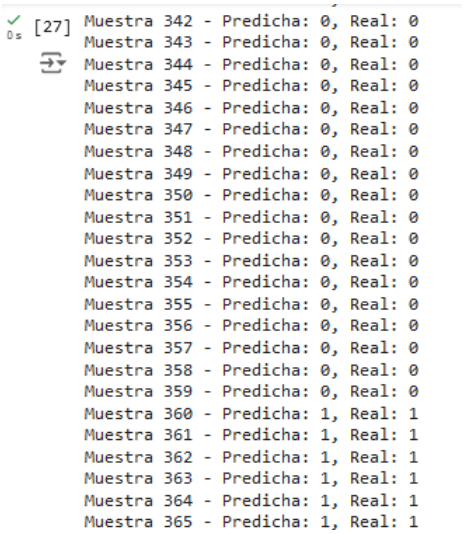


Figura 94. Muestras predichas de las clases 0 y 1 mediante la SVM

De la tabla 18 se puede concluir que los modelos de 9 capas MPL y máquina de soporte vectorial (SVM) presentan un desempeño eficiente en la tarea de detección de fallas de alta impedancia. Los porcentajes de 74.6% y 99.8 de predicciones correctas para la clase 0 y 1, respectivamente muestran que la red MPL puede ser implementada considerando la tasa de error que presenta ante muestras de clase 0. Por otro lado, el modelo de máquina de soporte vectorial presenta un 99.78 y 98.14% de predicciones correctas para la clase 0 y 1 respectivamente. Estos resultados subrayan la eficacia de la SVM en minimizar los errores de clasificación, lo que la hace particularmente adecuada para esta propuesta,

donde la precisión es crucial ya que un error conllevaría falsas alertas. En resumen, ambos modelos son seleccionados para ser implementados en una tarjeta Raspberry Pi 4.

Tabla 18. Porcentaje de predicción por parte de los modelos de IA ante nuevos datos

% Predicción ante nuevos datos						
Modelo	**Conjunto de datos 1**		**Conjunto de datos 2**		**Conjunto de datos 3**	
	Porcentaje de predicción		**Porcentaje de predicción**		**Porcentaje de predicción**	
	Clase 0 Crrct. – Inc.	**Clase 1 Crrct. – Inc.**	**Clase 0 Crrct. – Inc.**	**Clase 1 Crrct. – Inc.**	**Clase 0 Crrct. – Inc.**	**Clase 1 Crrct. – Inc.**
IA 1 (Red MPL)	100% 0%	62.6% 37.4%	0% 100%	100% 0%	12.6% 87.4%	100% 0%
IA 2 (Red MPL)	100% 0%	99.4% 0.6%	62.4% 37.6%	100% 0%	62.2% 37.8%	100% 0%
IA 3 (Red CNN)	41.40% 58.59%	63.80% 36.19%	0% 100%	54.49% 45.50%	12.39% 87.60%	100% 0%
IA 4 (Red CNN)	99.60% 0.39%	63.80% 36.19%	31.09% 68.90%	54.68% 45.31%	12.60% 87.39%	100% 0.0%
IA 5 (SVM)	100% 0%	96.93 % 3.06%	99.36% 0.63%	99.61 % 0.38%	100% 0%	97.89% 2.10%

De los modelos seleccionados se puede visualizar en la figura 95 que ambos presentan métricas eficientes al evaluar la predicción en el conjunto de evaluación utilizado del conjunto de datos utilizado para el entrenamiento de las diferentes redes neuronales, de igual forma los modelos restantes presentaron métricas relevantes, no obstante, su capacidad de generalización correcta ante nuevos datos se situó en el rango de 27.93% y 79% para datos de clase 0 y 1 respectivamente, de esta forma estos modelos presentaron una baja predicción para datos de operación normal.

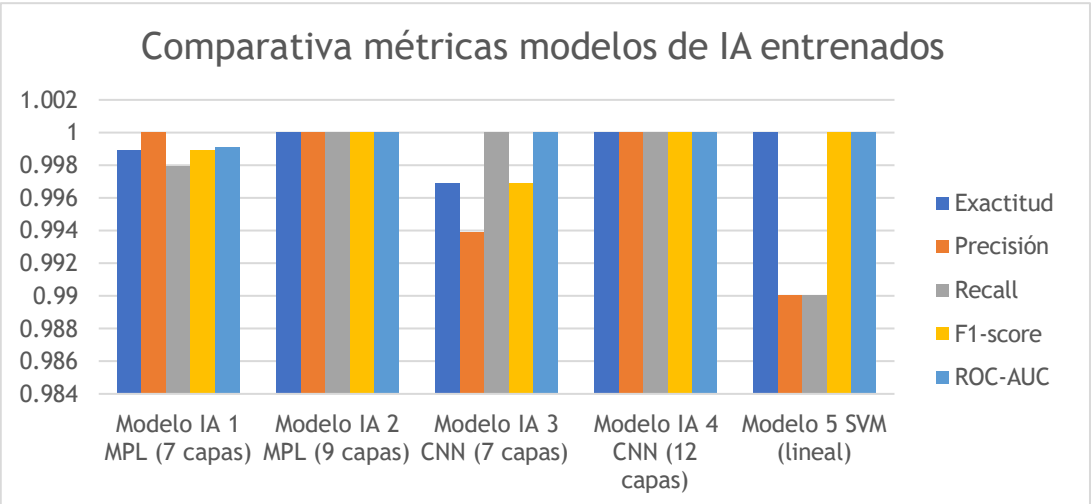


Figura 95. Comparativas métricas de los modelos de IA propuestos

3.5.4 Evaluación del modelo de inteligencia artificial con datos reales de clase 0 y 1

En esta sección, se evalúan los dos modelos de inteligencia artificial seleccionados (MPL de nueve capas y SVM) con la menor tasa de error utilizando datos reales extraídos de oscilografías de equipos de protección instalados en subestaciones eléctricas. Los datos reales contienen muestras de la onda sinusoidal de corriente nominal de un alimentador de distribución y onda de corriente eléctrica cuando se presenta una falla de alta impedancia. El objetivo es analizar la capacidad de ambos modelos para clasificar correctamente las respectivas clases (0 o 1) ante datos reales de campo, prestando especial atención a la detección de fallas de alta impedancia, las mismas que son difíciles de identificar debido a sus características sutiles y variables.

En la figura 96 se muestra una oscilografía extraída de un alimentador de distribución eléctrica, en la que se muestra las ondas sinusoidales de corriente eléctrica del sistema trifásico en condiciones normales, además, se ilustra una onda sinusoidal pura sin presencia de las características de asimetría ni no linealidades.

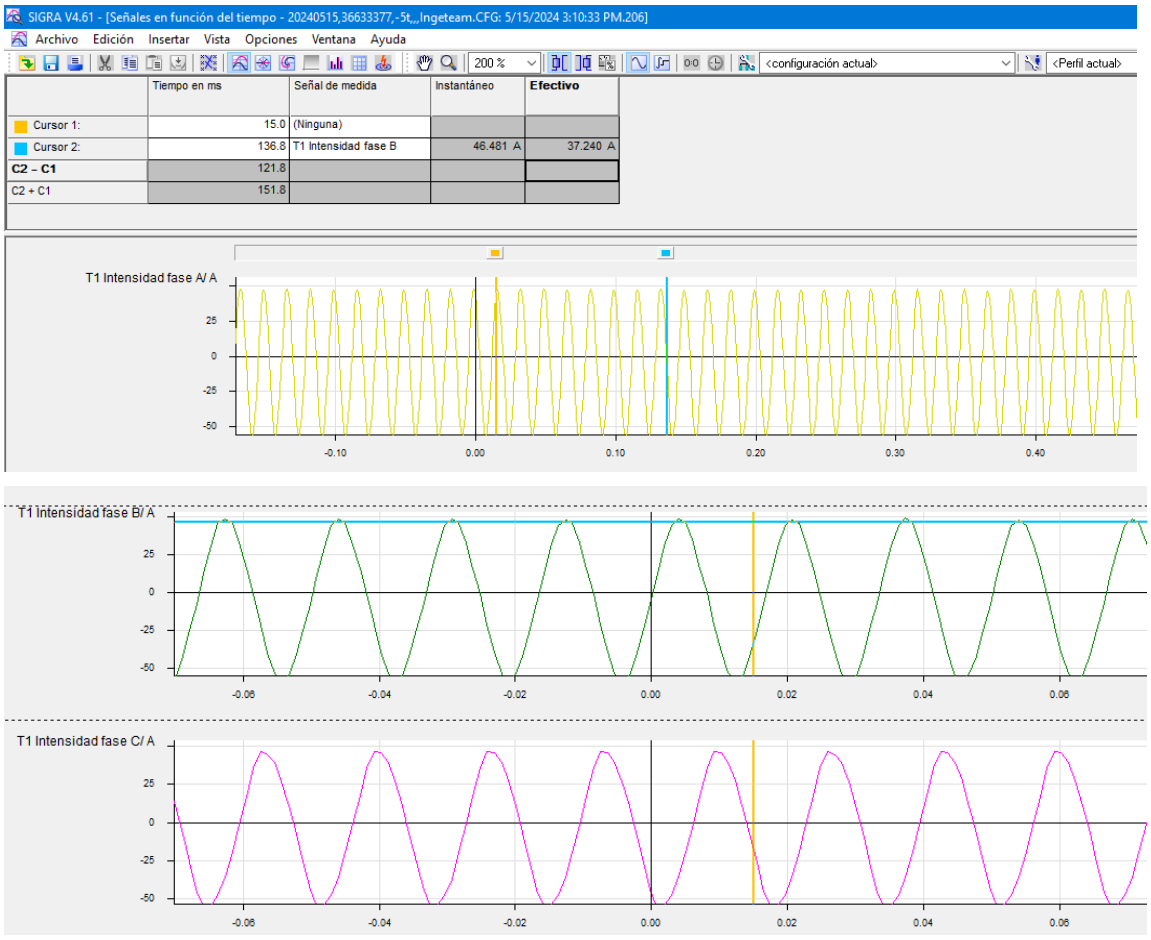


Figura 96. Datos reales de corriente nominal de un alimentador de distribución

Este conjunto de datos de la oscilografia de datos reales es exportado a un archivo csv para evaluar de esta forma las predicciones ejecutadas por los modelos de inteligencia artificial MPL y SVM respectivamente. En las siguientes figuras 97, 98 y 99 se visualizan los resultados obtenidos, ante una secuencia de datos reales de una onda de corriente eléctrica, en donde se ilustra que el modelo de inteligencia artificial MPL predice eficientemente las 40 muestras, clasificándolas como clase 0, con un 100% de precisión.

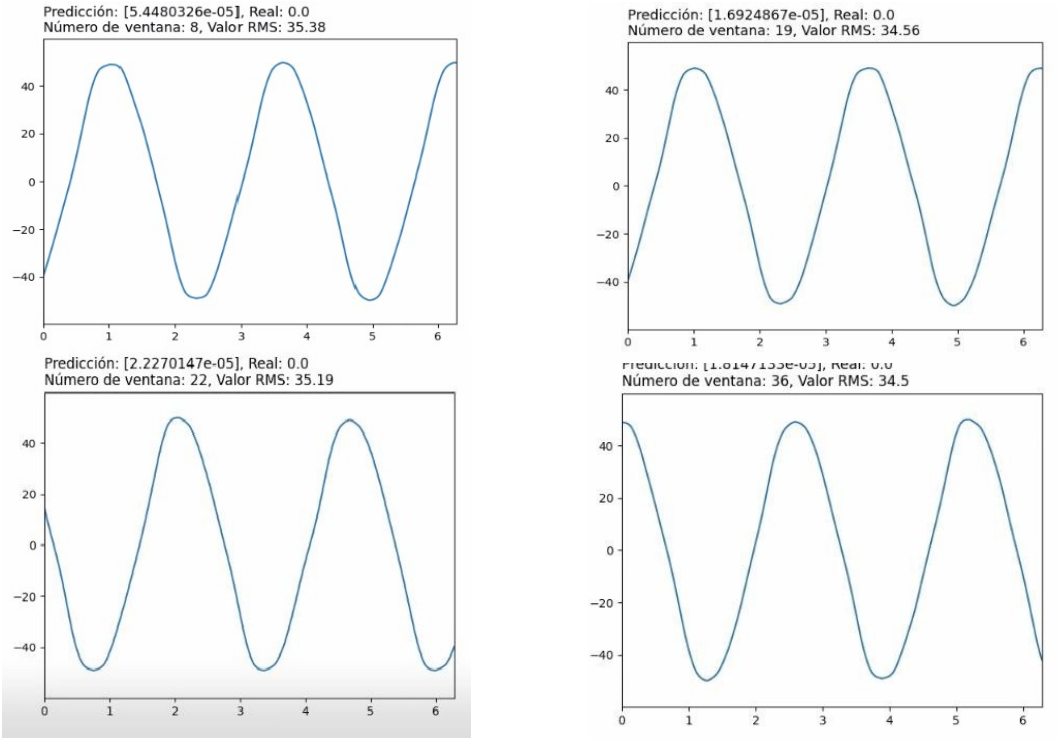


Figura 97. Predicciones de datos reales de Clase 0, red MPL de nueve capas

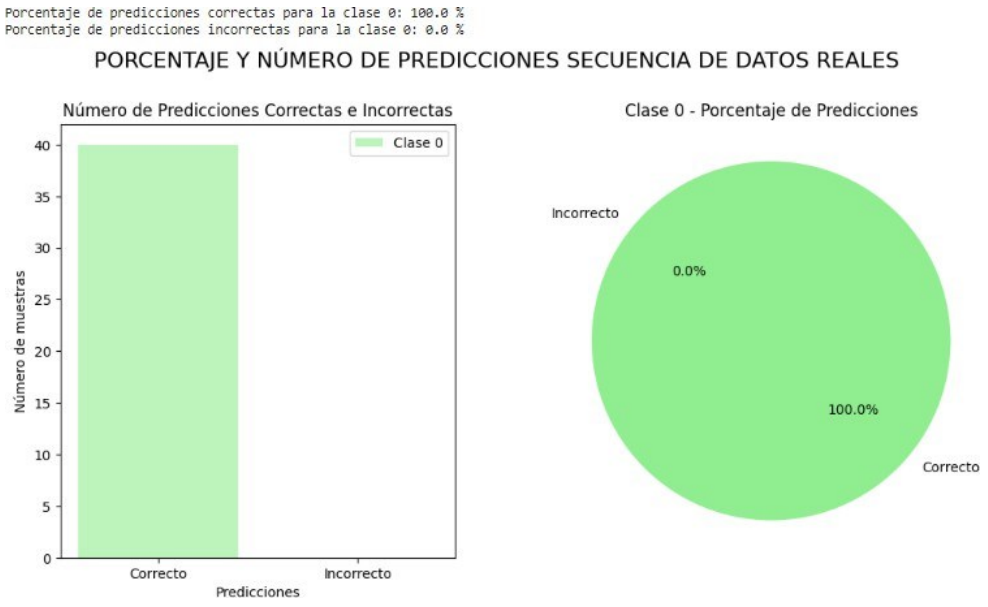
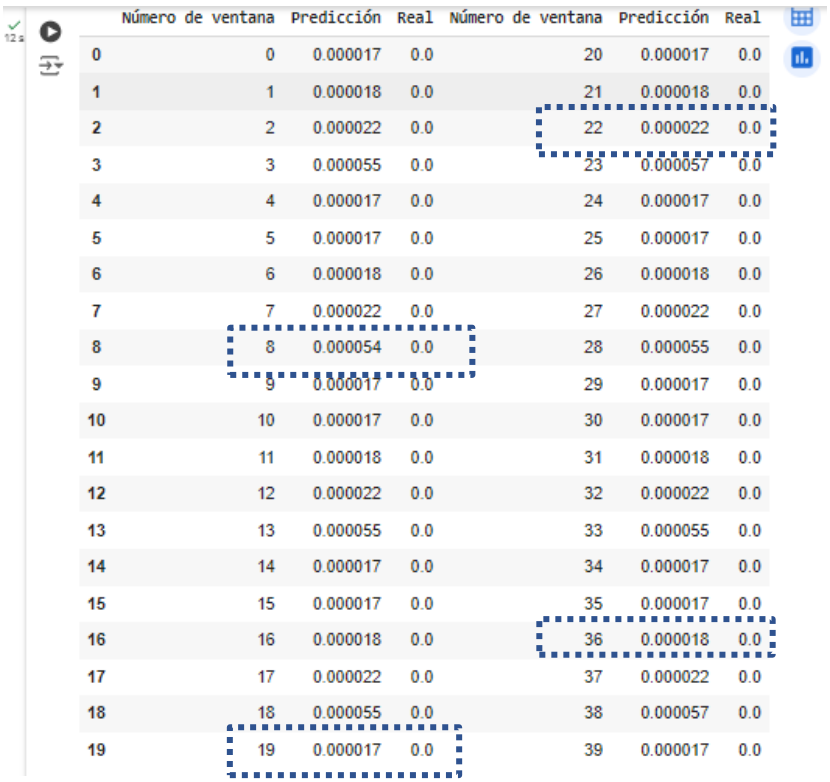


Figura 98. Gráfico de predicciones del modelo MPL



	Número de ventana	Predicción	Real	Número de ventana	Predicción	Real
0	0	0.000017	0.0	20	0.000017	0.0
1	1	0.000018	0.0	21	0.000018	0.0
2	2	0.000022	0.0	22	0.000022	0.0
3	3	0.000055	0.0	23	0.000057	0.0
4	4	0.000017	0.0	24	0.000017	0.0
5	5	0.000017	0.0	25	0.000017	0.0
6	6	0.000018	0.0	26	0.000018	0.0
7	7	0.000022	0.0	27	0.000022	0.0
8	8	0.000054	0.0	28	0.000055	0.0
9	9	0.000017	0.0	29	0.000017	0.0
10	10	0.000017	0.0	30	0.000017	0.0
11	11	0.000018	0.0	31	0.000018	0.0
12	12	0.000022	0.0	32	0.000022	0.0
13	13	0.000055	0.0	33	0.000055	0.0
14	14	0.000017	0.0	34	0.000017	0.0
15	15	0.000017	0.0	35	0.000017	0.0
16	16	0.000018	0.0	36	0.000018	0.0
17	17	0.000022	0.0	37	0.000022	0.0
18	18	0.000055	0.0	38	0.000057	0.0
19	19	0.000017	0.0	39	0.000017	0.0

Figura 99. Predicciones realizadas por el modelo de inteligencia artificial MPL

Los resultados de las figuras 100, 101 y 102 muestran las predicciones realizadas por el modelo de máquinas de vectores de soporte (SVM), ingresando datos reales de clase 0 (corriente eléctrica nominal o ausencia de falla). De manera general, se puede visualizar que el modelo obtuvo un desempeño mixto en las predicciones ejecutadas, ya que algunas muestras presentaron discrepancias entre la clase predicha y la real.

En las ventanas 1,2,6,7,11,12,16,17,21,22,26,27,31.32,36,37 del conjunto de datos reales ingresado, las predicciones realizadas por la SVM no se alinean, lo que indica un error de clasificación, mientras que para 24 ventanas restantes de datos el modelo predice correctamente. Sin embargo, existe un margen muy estrecho entre ambas predicciones y la generalización del modelo la cual dio como resultado un 60% de predicciones correctamente ante un 40% de manera incorrecta, lo cual puede marcar una desventaja al momento de implementar dicho modelo ya que cada tres muestras se visualizan que el modelo predice de manera incorrecta, lo que puede conllevar a falsas alarmas en un sistema crítico, como el eléctrico. En comparativa con la red neuronal MPL de nueve capas evaluada anteriormente presenta una precisión del 100% de precisión frente a un 60% de precisión por parte de la SVM.

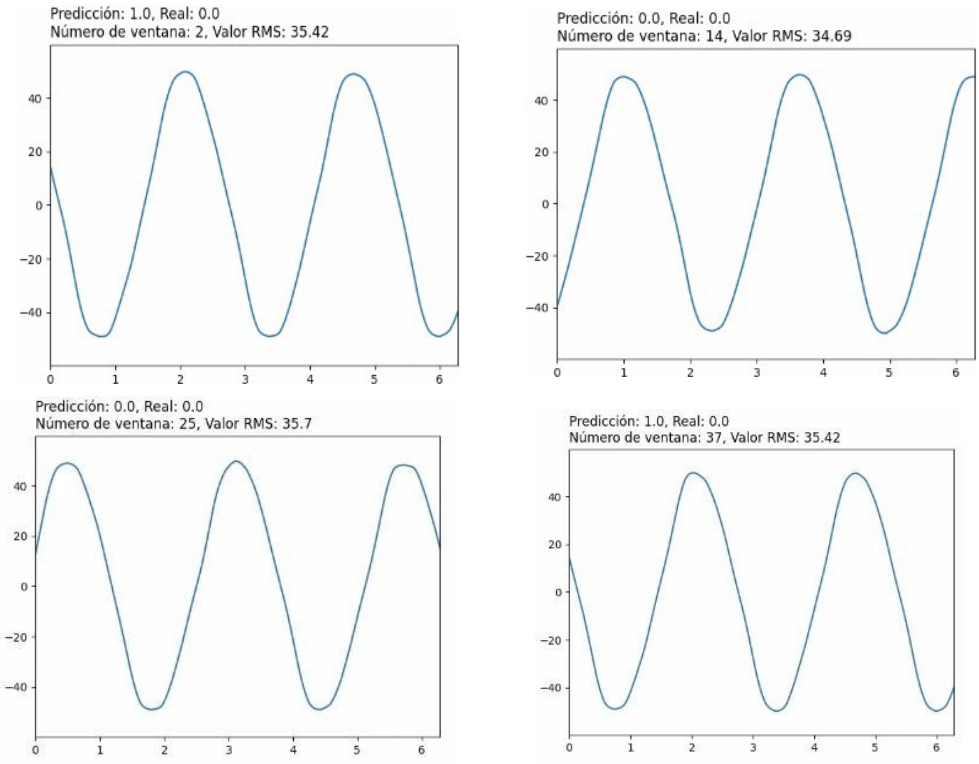


Figura 100. Predicciones de datos reales, Clase 0, algoritmo SVM

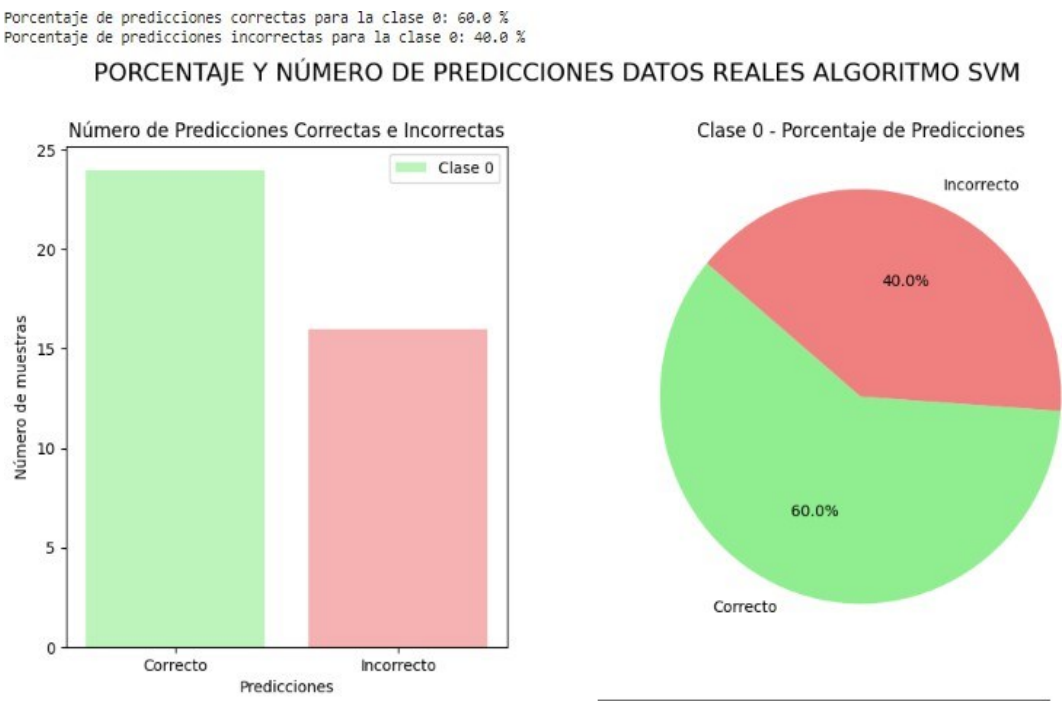


Figura 101. Gráfico de predicciones del algoritmo SVM

	Número de ventana	Predicción	Real	Número de ventana	Predicción	Real
0	0	0.0	0.0	20	0.0	0.0
1	1	1.0	0.0	21	1.0	0.0
2	2	1.0	0.0	22	1.0	0.0
3	3	0.0	0.0	23	0.0	0.0
4	4	0.0	0.0	24	0.0	0.0
5	5	0.0	0.0	25	0.0	0.0
6	6	1.0	0.0	26	1.0	0.0
7	7	1.0	0.0	27	1.0	0.0
8	8	0.0	0.0	28	0.0	0.0
9	9	0.0	0.0	29	0.0	0.0
10	10	0.0	0.0	30	0.0	0.0
11	11	1.0	0.0	31	1.0	0.0
12	12	1.0	0.0	32	1.0	0.0
13	13	0.0	0.0	33	0.0	0.0
14	14	0.0	0.0	34	0.0	0.0
15	15	0.0	0.0	35	0.0	0.0
16	16	1.0	0.0	36	1.0	0.0
17	17	1.0	0.0	37	1.0	0.0
18	18	0.0	0.0	38	0.0	0.0
19	19	0.0	0.0	39	0.0	0.0

Figura 102. Predicciones realizadas por el algoritmo SVM

Para corroborar la precisión y eficacia de los dos modelos MPL y SVM se despliega un nuevo conjunto de datos reales con un valor de amplitud pico incrementado (Ver figura 103), un mayor número de muestras y por ende un amplio número de ventanas, lo que permitirá analizar la precisión de datos de clase 0 por parte de los modelos de inteligencia artificial.

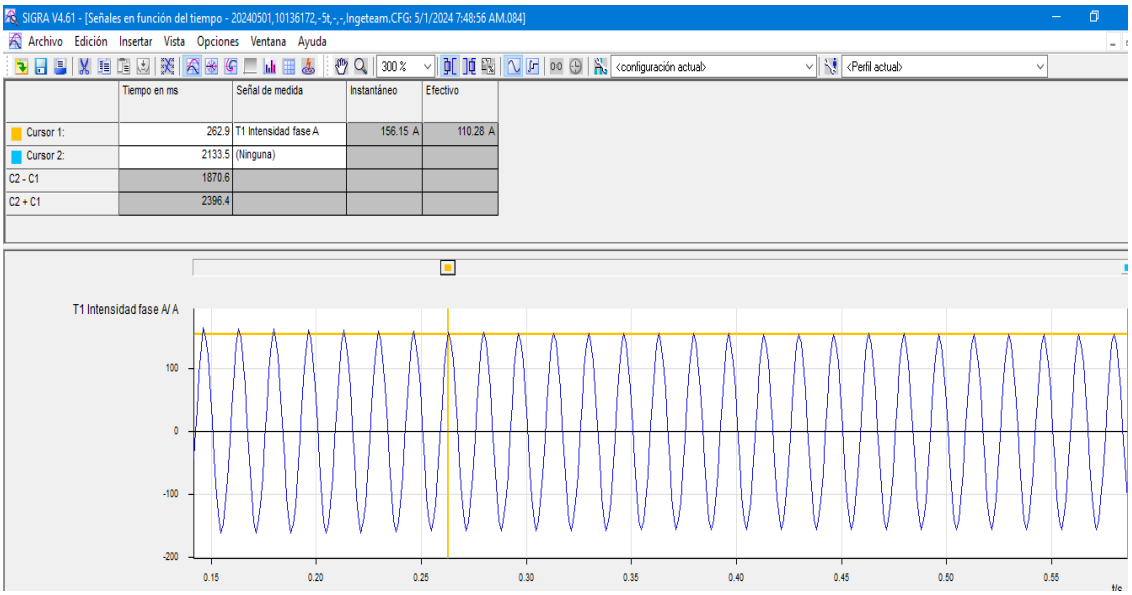


Figura 103. Corriente eléctrica real de flujo nominal de carga

Se analizan los resultados obtenidos ante un nuevo conjunto de datos, en donde se comprueba que la red MPL presenta un porcentaje de clasificación eficiente en comparativa con el algoritmo SVM. Un porcentaje del 100% frente a un 53.3% (Ver figuras 104 y 105) lo que demuestra que esta red MPL profunda es capaz de discriminar correctamente los eventos de clase 0, siendo capaz de distinguir entre un aumento de carga y flujo nominal de corriente eléctrica, no produciendo falsas detecciones o clasificaciones incorrectas (Ver figura 106).

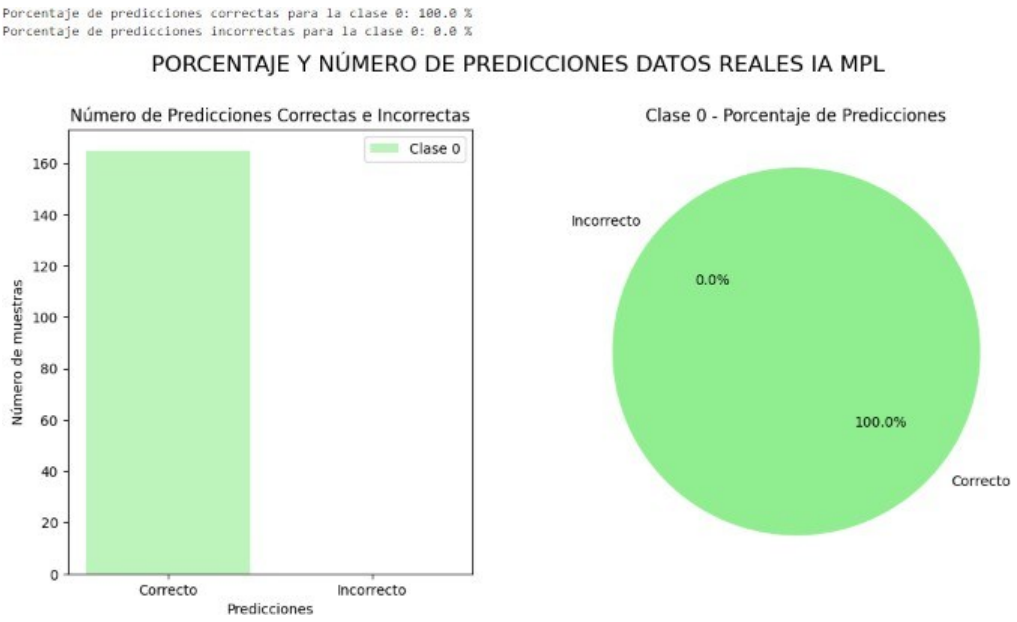


Figura 104. Porcentaje y predicciones antes datos reales de clase 0, MPL

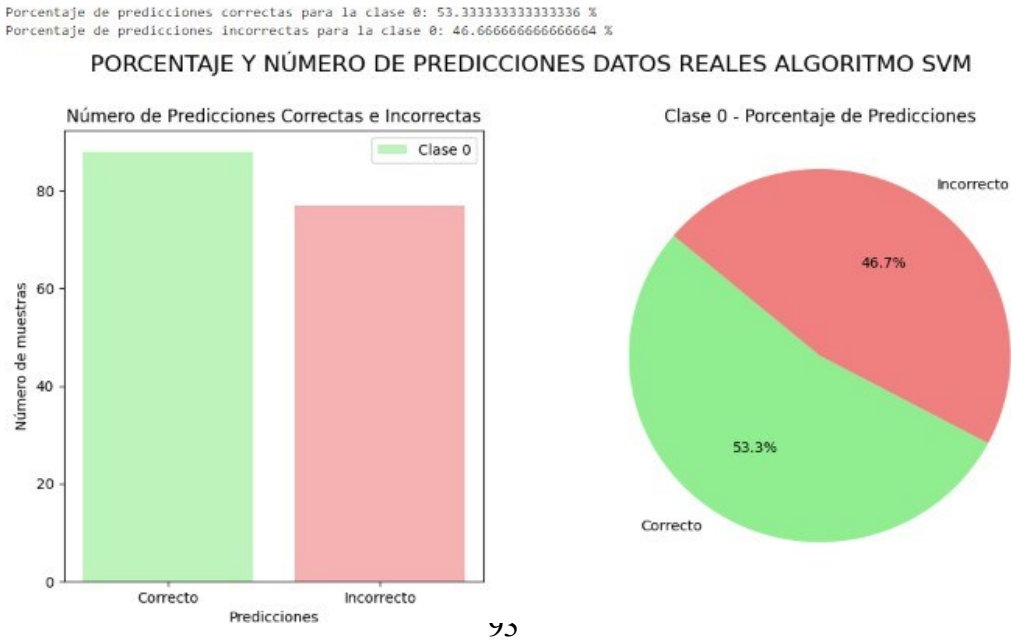


Figura 105. Porcentaje y predicciones ante datos reales de clase 0, SVM

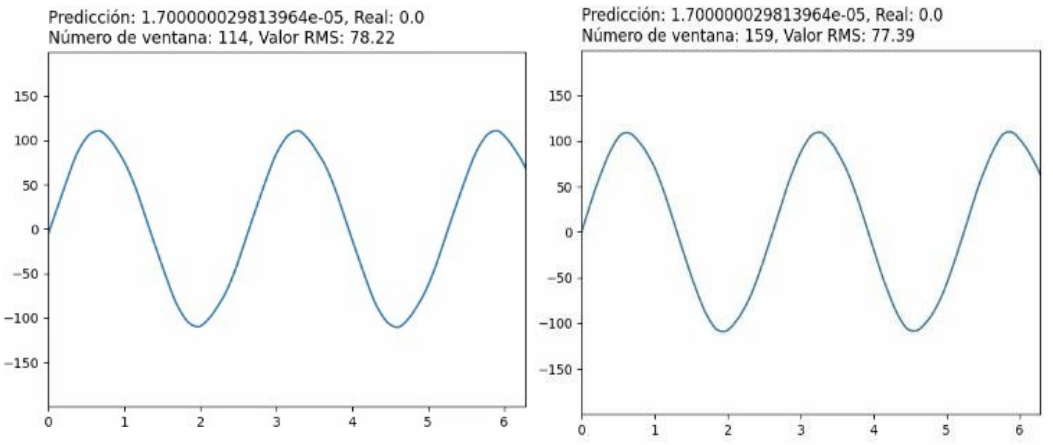


Figura 106. Detección eficiente de eventos de clase 0, MPL

Posteriormente los respectivos modelos seleccionados, se evalúan utilizando datos reales de un evento de alta impedancia presentado en un alimentador de distribución y recolectados de un relé de protección (Ver figura 107). Este conjunto de datos es procesado mediante una ventana de datos con un *overlap* de 1 ciclo, obteniéndose 33 ventanas de fallas de alta impedancia, las cuales son utilizadas como datos de entrada para evaluar los modelos de inteligencia artificial MPL y SVM. En la forma de onda real de una falla de alta impedancia, se puede observar la presencia de no linealidad en la onda de corriente eléctrica, así como la característica de *buildup*, donde se aprecia el incremento de la corriente eléctrica durante aproximadamente 4 ciclos hasta alcanzar un estado estable. En este caso la presencia de la característica de asimetría es mínima, lo que permite evaluar la la robustez del modelo para predecir la clase de los datos de entrada, como una corriente normal de carga o la presencia de un evento de alta impedancia en la red eléctrica de media tensión (13.8kV).

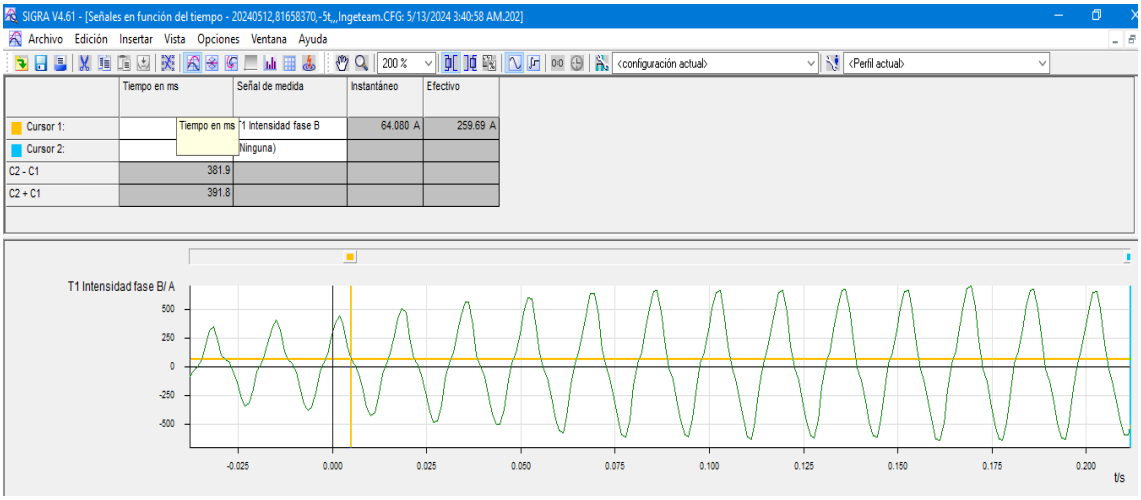


Figura 107. Conjuntos de datos reales de corriente eléctrica ante eventos de alta impedancia

Las predicciones obtenidas de ambos modelos se presentan en las Figuras 108, 109 y 110, donde los resultados muestran un desempeño notablemente superior de la red MPL, la cual, de las 33 muestras evaluadas, predice correctamente 26 y 7 erróneamente. En contraste, el algoritmo SVM logra 20 predicciones correctas y 13 incorrectas. Esto se traduce en un porcentaje de precisión del 78.79% y 60.61% para la red MPL y el algoritmo SVM respectivamente. La mayor precisión de la red MPL sugiere una capacidad robusta y eficiente para discernir eventos de fallas de alta impedancia, ya que, al emplear múltiples capas ocultas y técnicas de regularización, puede capturar las características complejas como la no linealidad y características sutiles como el *buildup* y las mínimas asimetrías en la onda de la corriente eléctrica. Esta ventaja se refleja en la capacidad de esta red neuronal para generalizar mejor en presencia de variaciones en los datos de la señal de entrada, subrayando el uso de inteligencia artificial para aplicaciones críticas como los sistemas de protección y diagnóstico de fallas en redes eléctricas de distribución.

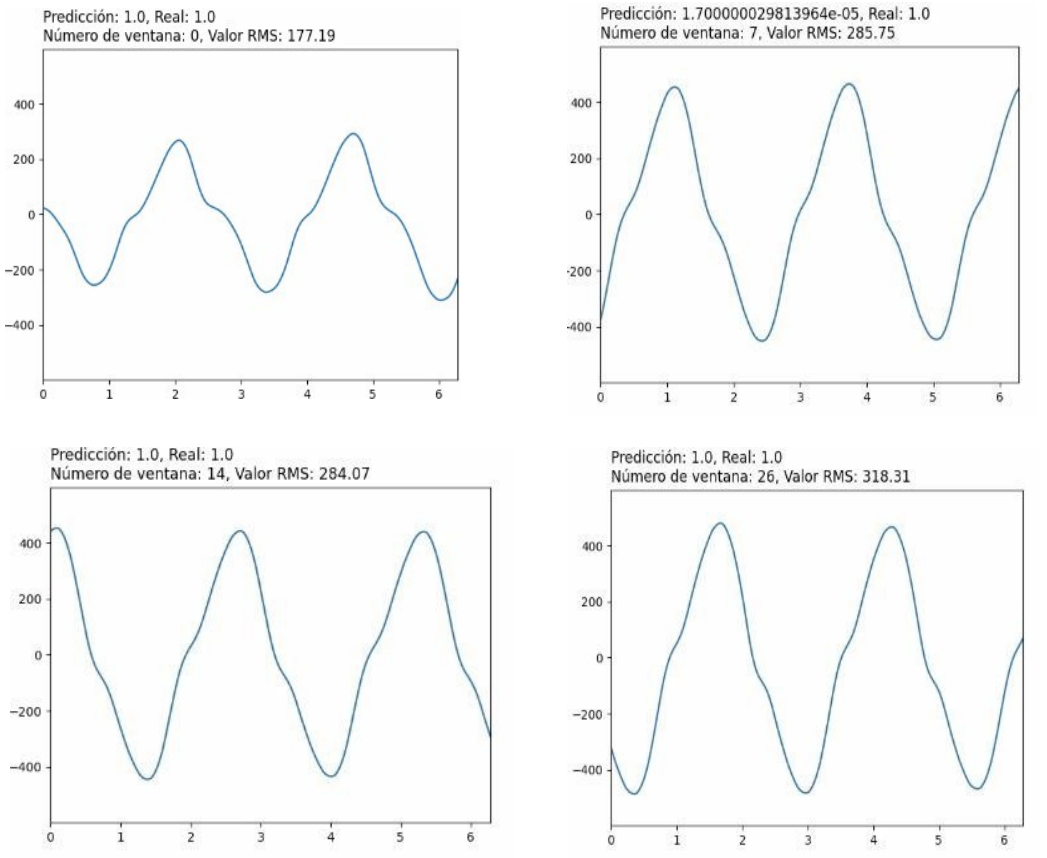


Figura 108. Predicciones de eventos de alta impedancia (clase 1), MPL

	Número de ventana	Predicción	Real	Número de ventana	Predicción	Real
0	0	1.000000	1.0	17.0	0.000017	1.0
1	1	1.000000	1.0	18.0	1.000000	1.0
2	2	0.000017	1.0	19.0	1.000000	1.0
3	3	1.000000	1.0	20.0	1.000000	1.0
4	4	1.000000	1.0	21.0	1.000000	1.0
5	5	1.000000	1.0	22.0	0.000017	1.0
6	6	1.000000	1.0	23.0	1.000000	1.0
7	7	0.000017	1.0	24.0	1.000000	1.0
8	8	1.000000	1.0	25.0	1.000000	1.0
9	9	1.000000	1.0	26.0	1.000000	1.0
10	10	1.000000	1.0	27.0	0.000017	1.0
11	11	1.000000	1.0	28.0	1.000000	1.0
12	12	0.000017	1.0	29.0	1.000000	1.0
13	13	1.000000	1.0	30.0	1.000000	1.0
14	14	1.000000	1.0	31.0	1.000000	1.0

Figura 109. Predicciones realizadas por la red MPL, ante datos reales de clase 1

	Número de ventana	Predicción	Real	Número de ventana	Predicción	Real
0	0	1.0	1.0	17.0	0.0	1.0
1	1	1.0	1.0	18.0	0.0	1.0
2	2	0.0	1.0	19.0	1.0	1.0
3	3	0.0	1.0	20.0	1.0	1.0
4	4	1.0	1.0	21.0	1.0	1.0
5	5	1.0	1.0	22.0	0.0	1.0
6	6	1.0	1.0	23.0	0.0	1.0
7	7	0.0	1.0	24.0	1.0	1.0
8	8	0.0	1.0	25.0	1.0	1.0
9	9	1.0	1.0	26.0	1.0	1.0
10	10	1.0	1.0	27.0	0.0	1.0
11	11	1.0	1.0	28.0	0.0	1.0
12	12	0.0	1.0	29.0	1.0	1.0
13	13	0.0	1.0	30.0	1.0	1.0
14	14	1.0	1.0	31.0	1.0	1.0
15	15	1.0	1.0	32.0	0.0	1.0

Figura 110. Predicciones realizadas por el algoritmo SVM, ante datos reales de clase 1

A manera de resumen, de los cinco modelos de inteligencia artificial propuestos, la red neuronal perceptrón de nueve capas presentó el mejor desempeño y robustez en la predicción de las clases 0 y 1, tanto con datos simulados como con datos reales. El uso de técnicas avanzadas de entrenamiento y regularización contribuyen a su robustez y superior desempeño, destacándose como la opción más viable entre los modelos adecuados, para aplicaciones críticas. En este caso la red MPL es implementada en un sistema embebido, como una tarjeta Raspberry Pi 4, para de esta forma evaluar su

capacidad de procesamiento y los tiempos de respuestas para emitir una alerta de falla de alta impedancia.

3.5.5 Despliegue del modelo de inteligencia artificial en un sistema embebido

Posteriormente luego de seleccionar la inteligencia artificial con la mínima tasa de error, se implementa dicho modelo en un sistema embebido como una Raspberry Pi 4, en esta tarjeta se realizó una codificación que permitió leer los datos reales y efectuar predicciones del respectivo modelo de IA en formato .pb (Ver Anexo 9). En el despliegue del modelo de inteligencia artificial se visualizó una respuesta eficiente de la tarjeta al efectuar las predicciones del conjunto de datos de entrada obteniendo tiempo de procesamiento en el orden de milisegundos, lo cual es beneficioso en aplicaciones críticas donde el tiempo para emitir una alarma es crítico, tal como lo es el sistema eléctrico de potencia. En este caso los tiempos de procesamiento obtenidos para la predicción de las clases 0 y 1 bordeo entre los 14.31ms a los 19.6ms, tiempo requerido por la unidad de procesamiento de la Raspberry Pi 4 para emitir una respuesta. Tomando en consideración la frecuencia de la señal eléctrica muestreada que es de 60Hz, la cual tiene un periodo de 16.66ms, se puede considerar los tiempos obtenidos como eficientes, ya que prácticamente el procesamiento de las ventanas de datos el sistema embebido lo efectúa en tiempo real en un promedio de 16.9ms para emitir una predicción de clase (Ver figuras 111, 112 y 113).

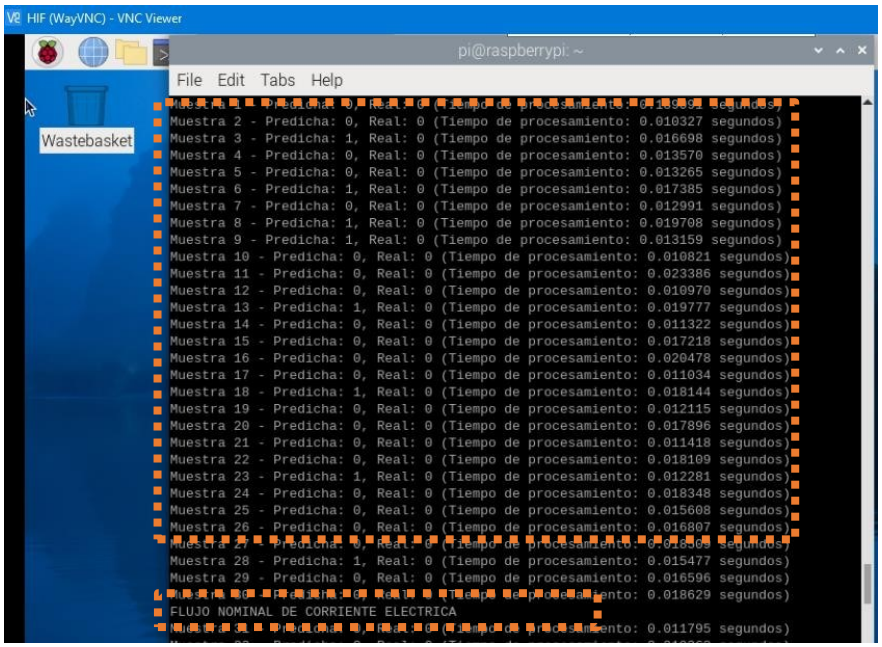


Figura 111. Predicciones de la clase 0 en el sistema embebido realizadas por la red MPL (30 muestras)

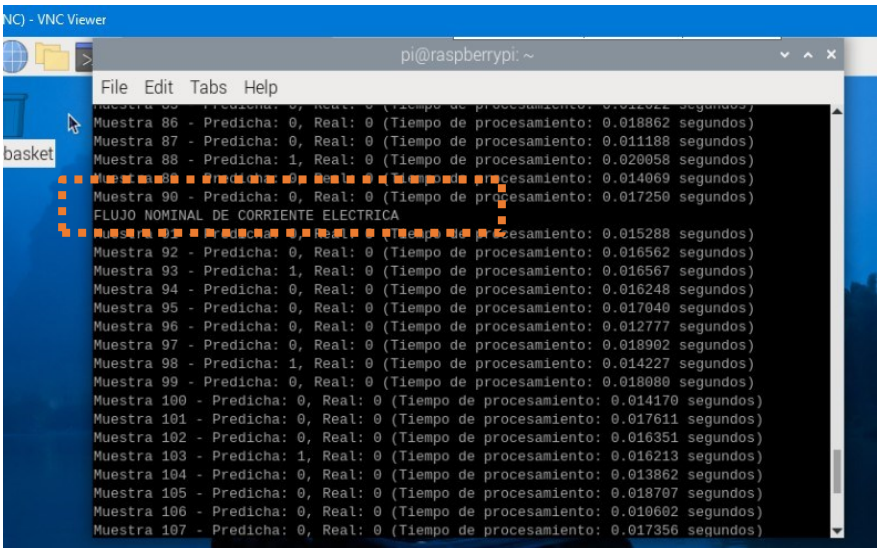


Figura 112. Predicciones de la clase 0 realizadas por la red MPL en el sistema embebido

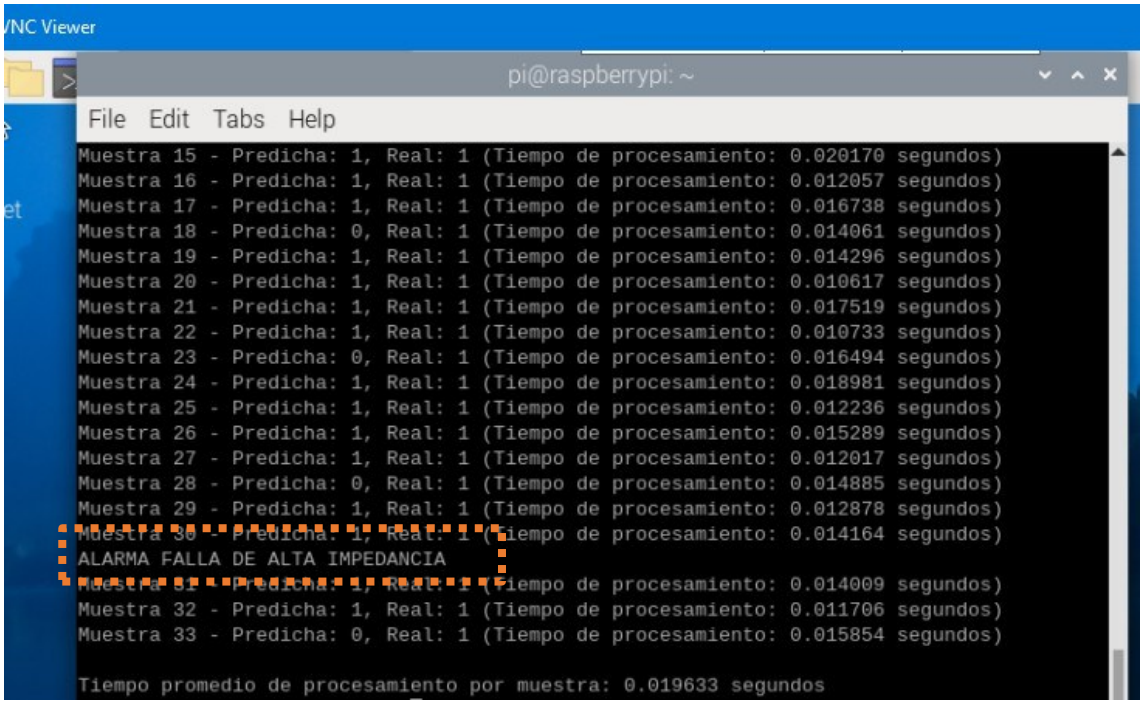


Figura 113. Predicciones de la clase 1 realizada por la red MPL en el sistema embebido

En la figura 114 se visualiza el rendimiento del sistema embebido al ejecutarse la red neuronal MPL, en la cual se muestra un consumo de recurso computacional del 42% por parte de la unidad central de procesamiento acompañada de un consumo de 575MB de memoria RAM. Estos valores reflejan una visión clara del nivel de carga que experimenta la tarjeta Raspberry Pi 4 al ejecutar tareas relacionadas con una arquitectura de

inteligencia artificial. Es importante destacar que el tiempo de procesamiento obtenido se puede interpretar como un procesamiento en “tiempo real”, lo cual puede contribuir al desempeño fiable del modelo desarrollado.

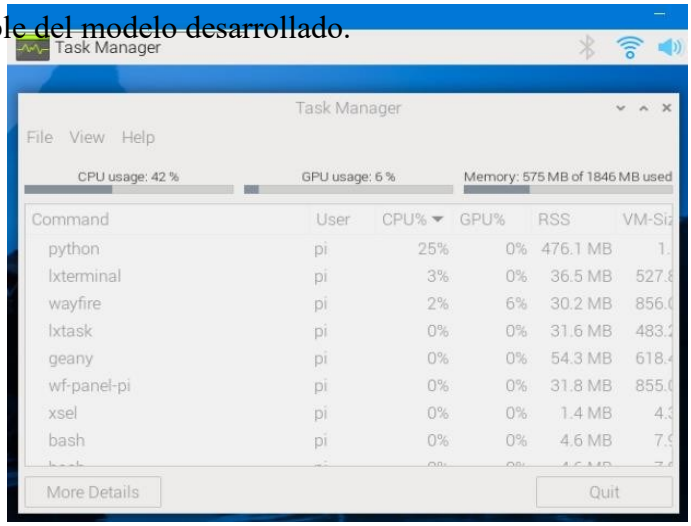


Figura 114. Rendimiento del sistema embebido al ejecutar el algoritmo de IA

Para evitar falsas alertas y mejorar la confiabilidad de predicción de los modelos de IA se implementó mediante codificación en el sistema embebido (Ver Anexo 9), la cual toma las predicciones de 30 conjuntos de datos y efectúa un promedio de la clase con mayor porcentaje de detección, emitiendo de esta forma una alerta. Si la clase mayoritaria presente es 0 se emite un mensaje indicando “Flujo nominal de corriente eléctrica”, caso contrario si existen muchas muestras que el modelo predijo como clase 1, se emite un mensaje “Alarma Falla de alta impedancia” (Ver figura 115), con lo cual se prevé alertar al operador para de esta forma, efectué las acciones pertinentes.

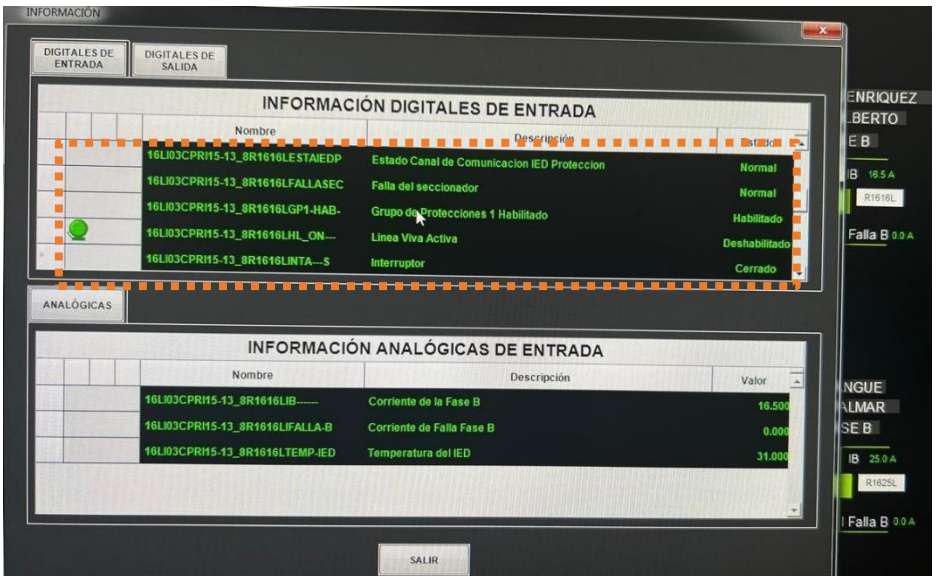


Figura 115. Propuesta de alarma en un sistema de supervisión remota

CONCLUSIONES

Mediante la herramienta MATLAB/Simulink se logró el modelado y la simulación de un alimentador de distribución eléctrica, utilizando los respectivos bloques de la librería especializada en sistemas de potencia. Se incluyeron bloques de fuente de tensión trifásica, transformador de potencia, líneas de distribución y las respectivas cargas electicas. Esto permitió simular el comportamiento del alimentador de la subestación eléctrica, ejecutar el flujo de carga y, en diferentes instantes de tiempo, inyectar eventos de falla de alta impedancia. Los datos fueron adquiridos a una velocidad de 500 muestras por ciclo, permitiendo capturar una amplia cantidad de información, incluyendo el flujo normal de la corriente eléctrica, aumentos de carga y fallas de alta impedancia de las diferentes superficies no conductoras propuestas en este trabajo. Con estos datos, se formó un *dataset* de entrenamiento que contuvo 5965 ventanas de datos, con lo que se consiguió un eficiente entrenamiento de las diferentes arquitecturas de IA propuestas.

Se propusieron cinco arquitecturas de inteligencia artificial empleando el entorno de Google Colab y el lenguaje de programación Python, dos redes Multi Layer Perceptron con configuración de siete y nueve capas, dos redes convolucionales de siete y doce capas y un algoritmo de máquina de vectores de soporte con *kernel* lineal y un factor de regularización de 0.01, en estas redes se logró implementar técnicas de optimización y regularización, logrando de esta forma asegurar un rendimiento óptimo de cada arquitectura de inteligencia artificial. Cada modelo fue exhaustivamente entrenado durante 150 épocas para la red MPL y 50 épocas para la red CNN en donde se empleó el 80% de las ventanas de datos para el entrenamiento mientras que el 20% de las ventanas de datos para validación.

Las métricas de evaluación obtenidas al entrenar los modelos de IA reflejaron que los modelos presentan un desempeño eficiente al ser evaluados con las muestras del 20% de los datos presentes en el conjunto de entrenamiento. Estas métricas eficientes se situaron por encima del 95%, lo que demuestra que los sistemas de IA son capaces de predecir patrones de fallas eléctricas, dependiendo de la calidad de la data empleada para el entrenamiento.

Al evaluar los modelos entrenados, ante conjuntos de datos no vistos durante el entrenamiento, se visualizó que la red MPL de siete capas y las redes convolucionales

presentaron errores de predicción en la clasificación de fallas, porcentajes muy bajos en el rango de 12% al 47% lo que indica que dichos modelos pudiesen encontrarse sobre-ajustados. En particular, se detectó que, al ingresar nuevos conjuntos de datos, las predicciones de salida tendían a ser consistentemente de la clase 1. ya que a cada nuevo conjunto de datos las predicciones de salida que emitían estos modelos eran de clase 1.

El rendimiento de la red MPL de nueve capas, se ubicó en el rango de 62.2% al 100% de predicciones correctas, sumamente superior al rendimiento de la primera red MPL donde los porcentajes de predicciones correctas abarcan desde el 12.6% al 100%, esta diferencia en las precisiones se logró debido a que la red de nueve capas posee dos capas ocultas totalmente de 2048 y 1024 neuronas las mismas que permiten capturar los patrones complejos presentes en las ventanas de datos de falla de alta impedancia.

Durante la puesta en operación de la red MPL de nueve capas en la tarjeta Raspberry Pi 4, se pudo visualizar que el consumo de recursos computacionales es notable, ya que la unidad central de procesamiento del sistema embebido realiza las respectivas operaciones para ejecutar las instrucciones matemáticas de la red neuronal, en este caso durante la ejecución del modelo se observó un 42% de consumo de la CPU y 575MB de memoria RAM, además de un tiempo de procesamiento de 16.9ms para emitir una etiqueta de predicción (Clase 0 o 1). Este tiempo de procesamiento obtenido refleja la viabilidad de este sistema embebido para la detección de este tipo de fallas en tiempo real.

Para minimizar la tasa de error en la predicción de las clase binaria y falsas alarmas, se implementó una lógica, en la que la red neuronal MPL evaluaba los datos de entrada cada 30 ventanas de datos (0.5 segundos), para de esta forma emitir una alerta emitir una alerta, dependiendo de la clase con mayor predicción en dicho intervalo de ventanas de datos, con esto se aumenta la confiabilidad del sistema de inteligencia artificial en este ambiente critico como son los sistemas eléctricos de potencia.

RECOMENDACIONES

En el presente trabajo se desarrolló un sistema capaz de detectar fallas de alta impedancia en una secuencia de datos provenientes de datos muestreados de una señal analógica de corriente, pudiendo emitir una alerta indicando el evento de alta impedancia, para trabajos futuros el modelo puede trabajar en conjunto con un sistema embebido para que enviase la señal de “trip” al relé de protección y de esta forma aislar de forma segura la falla, en coordinación con las demás protecciones del alimentador de energía.

Para obtener una data más eficiente se deben realizar ensayos con relés de protección, sistemas de inyección de corriente, ejecutando contactos con los conductores energizados y una superficie de alta impedancia, para de esta forma visualizar mediante la oscilografías del relé de protección la forma de onda producida al producirse esta falla.

REFERENCIAS

Abasi-obot, I., Kunya, A. B., Shehu, G. S., & Jibril, Y. (2023). High Impedance Fault Detection and Localization Using Fully-Connected Convolutional Neural Network: A Deep Learning Approach. *Nigerian Journal of Technological Development*, *20*(4). https://journal.njtd.com.ng/index.php/njtd/article/view/2143

Asensio, H. G., & Bowen, A. M. (s. f.). Inteligencia artificial.Redes neuronales y aplicaciones.

Bravo, R., & Pham, E. (2017). 12kV high impedance fault testing. *2017 IEEE Power & Energy Society General Meeting*, 1-5. https://doi.org/10.1109/PESGM.2017.8274061

Caicedo B, E. F., & López S, J. A. (2009). Una aproximación práctica a las redes neuronales artificiales (1.a ed.). Programa Editorial Universidad del Valle. https://doi.org/10.25100/peu.64

Choo, K.-K. R., & Dehghantanha, A. (2020). Handbook of Big Data Privacy. https://doi.org/10.1007/978-3-030-38557-6

Córdoba Guzmán, E. (2018). Desempeño de algoritmo wavelet para la detección de fallas de alta impedancia en redes del sistema de potencia con comunicación IEC 61850-9-2. https://repositorio.unal.edu.co/handle/unal/68609

Costa, F. B., Souza, B. A., Brito, N. S. D., Silva, J. A. C. B., & Santos, W. C. (2015). Real-Time Detection of Transients Induced by High-Impedance Faults Based on the Boundary Wavelet Transform. IEEE Transactions on Industry Applications, 51(6), 5312-5323. https://doi.org/10.1109/TIA.2015.2434993

Dery, L., Nachman, B., Rubbo, F., & Schwartzman, A. (2017). Weakly Supervised Classification in High Energy Physics. Journal of High Energy Physics, 2017. https://doi.org/10.1007/JHEP05(2017)145

Fan, R., & Yin, T. (2019). Convolutional Neural Network and Transfer Learning for High Impedance Fault Detection (arXiv:1904.08863). arXiv. https://doi.org/10.48550/arXiv.1904.08863

Ghaderi, A., Mohammadpour, H. A., Ginn, H. L., & Shin, Y.-J. (2015). High-Impedance Fault Detection in the Distribution Network Using the Time-Frequency-Based Algorithm. IEEE Transactions on Power Delivery, 30(3), 1260-1268. https://doi.org/10.1109/TPWRD.2014.2361207

Gómez Quesada, F. J., Fernández Graciani, M. A., López Bonal, M. T., & Alonso Díaz-Mata, M. (1994). Aprendizaje con redes neuronales artificiales. Ensayos: Revista de la Facultad de Educación de Albacete, 9, 169-180.

Hao, B., Yuxin, L., Jieyi, L., Hongwen, L., Yipeng, L., & Ruigui, L. (2023). High impedance fault detection device based on edge artificial intelligence. *Energy Reports*, *9*, 546-550. https://doi.org/10.1016/j.egyr.2023.09.168

Haykin, S. S. (2009). Neural networks and learning machines (3rd ed). Prentice Hall.

Joga, S. R. K., Sinha, P., & Maharana, M. K. (2019). Artificial Intelligence in classifying high impedance faults in Electrical Power Distribution System. En Proceedings of the International Conference on Recent Trends in Computing, Communication and Networking Technologies (ICRTCCNT'19), Kings Engineering College, October 18-19, 2019, Chennai, Tamilnadu, India. Bhubaneswar, India: School of Electrical Engineering, KIIT Deemed to be University. Disponible en: https://www.researchgate.net/publication/352896250_Artificial_Intelligence_in_classifying_high_impedance_faults_in_Electrical_Power_Distribution_System

Kujur, A. P., & Biswal, T. (2017). Detection of high impedance fault in distribution system considering distributed generation. 2017 International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), 406-410. https://doi.org/10.1109/ICIMIA.2017.7975646

Lu, S., & Lysecky, R. (2017). Time and Sequence Integrated Runtime Anomaly Detection for Embedded Systems. ACM Transactions on Embedded Computing Systems, 17, 1-27. https://doi.org/10.1145/3122785

Milioudis, A., Andreou, G., & Labridis, D. (2012). Enhanced Protection Scheme for Smart Grids Using Power Line Communications Techniques—Part II: Location of High Impedance Fault Position. IEEE Transactions on Smart Grid, 3, 1631-1640. https://doi.org/10.1109/TSG.2012.2208988

Moreno, L., Garrido, S., & Copaci, D. (s. f.). Fundamentos de las Redes Neuronales L6.

Moloi, K., Jordaan, J. A., & Hamam, Y. (2018). Support Vector Machine Based Method for High Impedance Fault Diagnosis in Power Distribution Networks. En H. Yin, D. Camacho, P. Novais, & A. J. Tallón-Ballesteros (Eds.), Intelligent Data Engineering and Automated Learning – IDEAL 2018 (pp. 9-16). Springer International Publishing. https://doi.org/10.1007/978-3-030-03493-1_2

Moloi, K., Jordaan, J. A., & Hamam, Y. (2019). A hybrid method for high impedance fault classification and detection. *2019 Southern African Universities Power Engineering Conference/Robotics and Mechatronics/Pattern Recognition Association of South Africa (SAUPEC/RobMech/PRASA)*, 548-552. https://doi.org/10.1109/RoboMech.2019.8704765

Oza, B., Nair, N.-K., Mehta, R., & Makwana, V. (2010). Power System Protection and Switchgear. New Delhi: McGraw Hill Education.

Pastor Ruiz, V. (2021, septiembre). Desarrollo de redes neuronales para resolución de problemas de estructuras [Info:eu-repo/semantics/bachelorThesis]. E.T.S.I. Industriales (UPM). https://oa.upm.es/68683/

Powers, D. (2008). Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation. Mach. Learn. Technol., 2.

Rai, K. (2021). Deep Learning for High-Impedance Fault Detection and Classification. *Electronic Thesis and Dissertation Repository*. https://ir.lib.uwo.ca/etd/7962

Ramiro, M. L. J. (s. f.). Óptima coordinación de protecciones para despejar fallas de alta impedancia en redes de distribución eléctrica.

Rouhiainen, L. (2018). Inteligencia artificial: 101 cosas que debes saber hoy sobre nuestro futuro. Alienta Editorial.

Ruiz, V. P., & Garcıa, D. P. (s. f.-a). DESARROLLO DE REDES NEURONALES PARA RESOLUCIO´ N DE PROBLEMAS DE ESTRUCTURAS.

Sánchez, F., & Alberto, L. (2023). Electrónica y sistemas embebidos Una visión a nivel técnico y tecnológico. Centro de Investigación y Desarrollo Ecuador. http://repositorio.cidecuador.org/jspui/handle/123456789/2398

Santos, W., Souza, B., Brito, N., Costa, F., & Paes Junior, M. (2013). High Impedance Faults: From Field Tests to Modeling. Journal of Control, 24. https://doi.org/10.1007/s40313-013-0072-8

Sharaf, A. M., Snider, L. A., & Debnath, K. (1993). A neural network based relaying scheme for distribution system high impedance fault detection (p. 324). https://doi.org/10.1109/ANNES.1993.323013

Sokolova, M., Japkowicz, N., & Szpakowicz, S. (2006). Beyond Accuracy, F-Score and ROC: A Family of Discriminant Measures for Performance Evaluation. En A. Sattar & B. Kang (Eds.), AI 2006: Advances in Artificial Intelligence (pp. 1015-1021). Springer. https://doi.org/10.1007/11941439_114

Steen, D. (2020, septiembre 14). Understanding the ROC Curve and AUC. Medium. https://towardsdatascience.com/understanding-the-roc-curve-and-auc-dd4f9a192ecb

Sekar, K., & Mohanty, N. (2017). Combined Mathematical Morphology and Data Mining Based High Impedance Fault Detection. Energy Procedia, 117, 417-423. https://doi.org/10.1016/j.egypro.2017.05.161

Shihabudheen, K. V., Kunju, B., Ahammed, I., Guruvarurappan, A., Jose, J., Keerthana, D., & Revathi, P. B. (2019). Detection of High Impedance Fault using Machine Learning Techniques. TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON), 2117-2122. https://doi.org/10.1109/TENCON.2019.8929365

Shihabudheen, K. V., Kunju, B., Ahammed, I., Guruvarurappan, A., Jose, J., Keerthana, D., & Revathi, P. B. (2019). Detection of High Impedance Fault using Machine Learning Techniques. *TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON)*, 2117-2122. https://doi.org/10.1109/TENCON.2019.8929365

Sirojan, T., Lu, S., Phung, B. T., Zhang, D., & Ambikairajah, E. (2018). High Impedance Fault Detection by Convolutional Deep Neural Network. *2018 IEEE International Conference on High Voltage Engineering and Application (ICHVE)*, 1-4. https://doi.org/10.1109/ICHVE.2018.8642080

Theron, J. C. J., Pal, A., & Varghese, A. (2018). Tutorial on high impedance fault detection. 2018 71st Annual Conference for Protective Relay Engineers (CPRE), 1-23. https://doi.org/10.1109/CPRE.2018.8349833

Vieira, F. L., Filho, J. M. C., Silveira, P. M., Guerrero, C. A. V., & Leite, M. P. (2018). High impedance fault detection and location in distribution networks using smart meters. *2018 18th International Conference on Harmonics and Quality of Power (ICHQP)*, 1-6. https://doi.org/10.1109/ICHQP.2018.8378825

Veerasamy, V., Abdul Wahab, N.I., Ramachandran, R., & et al. (2019). High-impedance fault detection in medium-voltage distribution network using computational intelligence-based classifiers. Neural Computing & Applications, 31(12), 9127–9143. https://doi.org/10.1007/s00521-019-04445-w

Wang, S., & Chen, H. (2019). A novel deep learning method for the classification of power quality disturbances using deep convolutional neural network. Applied Energy, 235, 1126-1140. https://doi.org/10.1016/j.apenergy.2018.09.160

Wang, S., & Dehghanian, P. (2020). On the Use of Artificial Intelligence for High Impedance Fault Detection and Electrical Safety. *IEEE Transactions on Industry Applications*, *56*(6), 7208-7216. https://doi.org/10.1109/TIA.2020.3017698

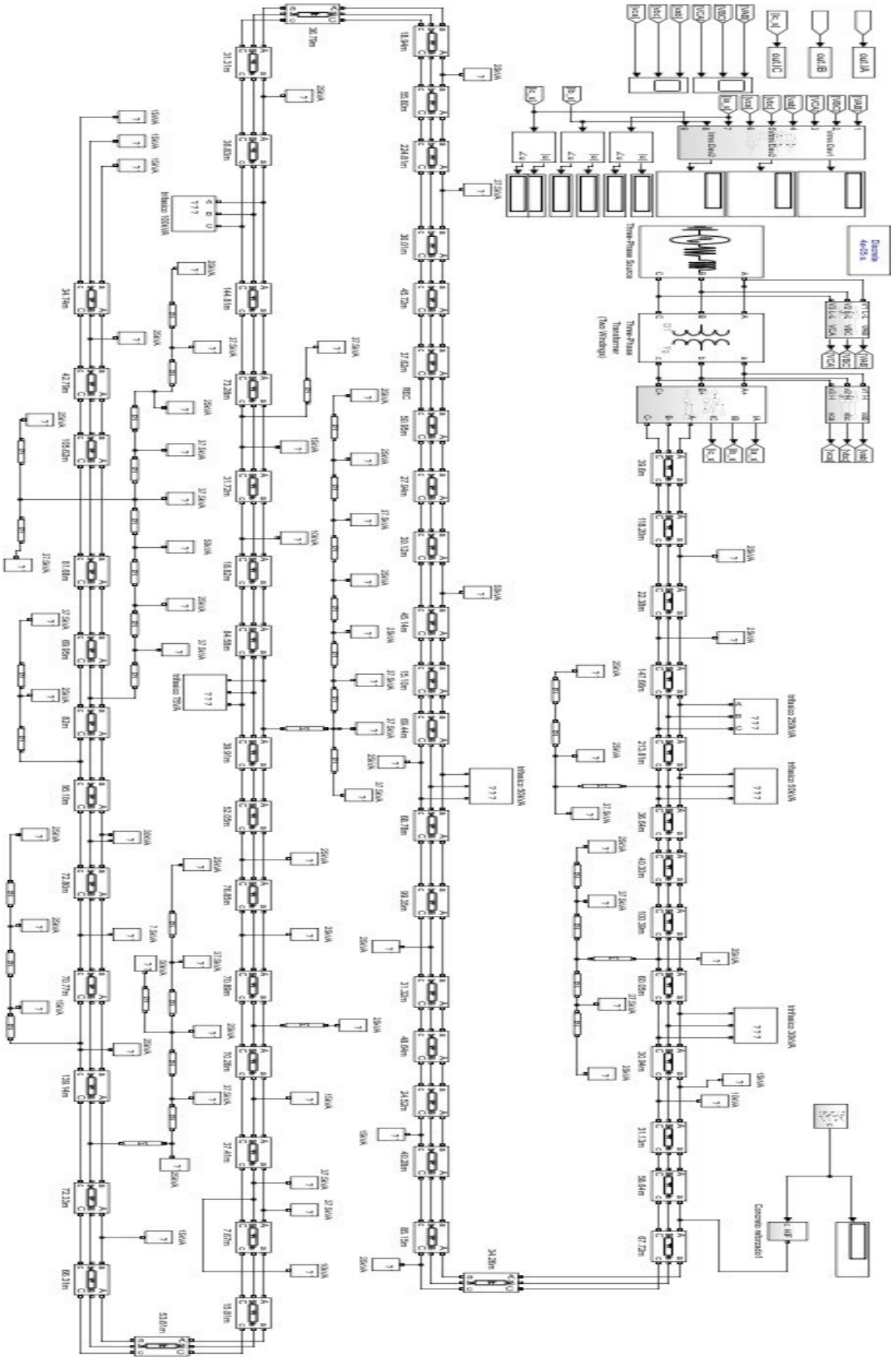
Wavelet transform based relay algorithm for the detection of stochastic high impedance faults—ScienceDirect. (s. f.). Recuperado 15 de mayo de 2024, de https://www.sciencedirect.com/science/article/abs/pii/S0378779605002816

Yebra, J. (2010). Sistemas Eléctricos de Distribución. Barcelona: REVERTÉ.

Zamanan, N., & Sykulski, J. K. (2006). Modelling arcing high impedances faults in relation to the physical processes in the electric arc. WSEAS Transactions on Power Systems, 1.

ANEXOS

Anexo 1. Diagrama esquemático, del alimentador eléctrico desarrollado en MATLAB/Simulink



Anexo 2. Codificación en Google Colab para el entrenamiento de la Red MPL 1

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout,
BatchNormalization
from tensorflow.keras.layers import Dense
from google.colab import drive
drive.mount('/content/drive')
# Cargar los datos desde los archivos CSV
data_corrientes =
np.loadtxt('/content/drive/MyDrive/DATA_IA/CORRIENTES_NORMAL_y_FALL
A.csv', delimiter=',')
data_salida =
np.loadtxt('/content/drive/MyDrive/DATA_IA/SALIDA_CLASES.csv',
delimiter=',')
# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test =
train_test_split(data_corrientes, data_salida, test_size=0.2,
random_state=42)
#RED MPL 1
model = Sequential([
    Dense(512, activation='relu', input_shape=(1000,),
kernel_regularizer='l2'),
    Dense(256, activation='relu', kernel_regularizer='l2'),
    Dense(128, activation='relu', kernel_regularizer='l2'),
    Dense(64, activation='relu', kernel_regularizer='l2'),
    Dense(32, activation='relu', kernel_regularizer='l2'),
    Dense(16, activation='relu', kernel_regularizer='l2'),
    Dense(1, activation='sigmoid')
])
# Imprimir el resumen del modelo
model.summary()
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
# Entrenar el modelo MPL
history = model.fit(X_train, y_train, epochs=150, batch_size=32,
validation_data=(X_test, y_test))
# Obtener las métricas de entrenamiento
training_loss = history.history['loss']
validation_loss = history.history['val_loss']
training_accuracy = history.history['accuracy']
validation_accuracy = history.history['val_accuracy']
```

```
epochs = range(1, len(training_loss) + 1)

plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.plot(epochs, training_loss, label='Pérdida de Entrenamiento')
plt.plot(epochs, validation_loss, label='Pérdida de Validación')
plt.title('Pérdida de Entrenamiento y Validación')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

# Plotear la curva de precisión durante el entrenamiento
plt.subplot(1, 2, 2)
plt.plot(epochs, training_accuracy, label='Exactitud de
Entrenamiento')
plt.plot(epochs, validation_accuracy, label='Exactitud de
Validación')
plt.title('Exactitud de Entrenamiento y Validación')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

plt.tight_layout()
plt.show()
# Evaluar el rendimiento del modelo
loss, accuracy = model.evaluate(X_test, y_test)
print('Loss:', loss)
print('Accuracy:', accuracy)
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, roc_auc_score, confusion_matrix
# Predicciones del modelo en el conjunto de prueba
y_pred = model.predict(X_test)
y_pred_binary = (y_pred > 0.5).astype(int)

# Calcular métricas
accuracy = accuracy_score(y_test, y_pred_binary)
precision = precision_score(y_test, y_pred_binary)
recall = recall_score(y_test, y_pred_binary)
f1 = f1_score(y_test, y_pred_binary)
roc_auc = roc_auc_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred_binary)

print('Accuracy:', accuracy)
print('Precision:', precision)
print('Recall:', recall)
print('F1-score:', f1)
print('ROC-AUC Score:', roc_auc)
print('Confusion Matrix:')
print(conf_matrix)
```

```
import tensorflow as tf
# Guardar el modelo entrenado
tf.saved_model.save(model,
'/content/drive/MyDrive/IA_ENTRENAMIENTO/MODELOMPL_1_IA_HIF')
```

Anexo 3. Codificación en Google Colab para el entrenamiento de la Red MPL 2

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout,
BatchNormalization
from tensorflow.keras.layers import Dense
from google.colab import drive
drive.mount('/content/drive')
# Cargar los datos desde los archivos CSV
data_corrientes =
np.loadtxt('/content/drive/MyDrive/DATA_IA/CORRIENTES_NORMAL_y_FALL
A.csv', delimiter=',')
data_salida =
np.loadtxt('/content/drive/MyDrive/DATA_IA/SALIDA_CLASES.csv',
delimiter=',')
# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test =
train_test_split(data_corrientes, data_salida, test_size=0.2,
random_state=42)
#Modelo Red MPL 2
model = Sequential([
    Dense(2048, activation='relu', input_shape=(1000,),
kernel_regularizer='l2'),
    Dense(1024, activation='relu', kernel_regularizer='l2'),
    Dense(512, activation='relu', kernel_regularizer='l2'),
    Dense(256, activation='relu', kernel_regularizer='l2'),
    Dense(128, activation='relu', kernel_regularizer='l2'),
    Dense(64, activation='relu', kernel_regularizer='l2'),
    Dense(32, activation='relu', kernel_regularizer='l2'),
    Dense(16, activation='relu', kernel_regularizer='l2'),
    Dense(1, activation='sigmoid')
])
# Imprimir el resumen del modelo
model.summary()
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
# Entrenar el modelo MPL
```

```
history = model.fit(X_train, y_train, epochs=150, batch_size=32,
validation_data=(X_test, y_test))
import tensorflow as tf
# Guardar el modelo entrenado
tf.saved_model.save(model,
'/content/drive/MyDrive/IA_ENTRENAMIENTO/MODELOMPL_2_IA_HIF')
```

Anexo 4. Codificación en Google Colab para el entrenamiento de la Red CNN 1

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout,
BatchNormalization
from tensorflow.keras.layers import Dense
from google.colab import drive
from keras.models import Sequential
from keras.layers import Conv1D, MaxPooling1D, Flatten, Dense
drive.mount('/content/drive')
# Cargar los datos desde los archivos CSV
data_corrientes =
np.loadtxt('/content/drive/MyDrive/DATA_IA/CORRIENTES_NORMAL_y_FALL
A.csv', delimiter=',')
data_salida =
np.loadtxt('/content/drive/MyDrive/DATA_IA/SALIDA_CLASES.csv',
delimiter=',')
# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test =
train_test_split(data_corrientes, data_salida, test_size=0.2,
random_state=42)
# Definir el modelo
model = Sequential()
# Agregar capas convolucionales
model.add(Conv1D(64, kernel_size=3, activation='relu',
input_shape=(1000, 1)))
model.add(Conv1D(128, kernel_size=3, activation='relu'))
model.add(Conv1D(256, kernel_size=3, activation='relu'))
model.add(MaxPooling1D(pool_size=2))
# Aplanar los datos
model.add(Flatten())
# Agregar capas densas
model.add(Dense(256, activation='relu',kernel_regularizer='l2'))
model.add(Dense(128, activation='relu',kernel_regularizer='l2'))
model.add(Dense(64, activation='relu',kernel_regularizer='l2'))
model.add(Dense(1, activation='sigmoid'))
```

```
# Imprimir el resumen del modelo
model.summary()
# Compilar el modelo
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
# Entrenar el modelo
history=model.fit(X_train, y_train, epochs=50, batch_size=32,
validation_data=(X_test, y_test))
import tensorflow as tf
# Guardar el modelo entrenado
tf.saved_model.save(model,
'/content/drive/MyDrive/IA_ENTRENAMIENTO/MODELOCNN_1_IA_HIF')
```

Anexo 5. Codificación en Google Colab para el entrenamiento de la Red CNN 2

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout,
BatchNormalization
from tensorflow.keras.layers import Dense
from google.colab import drive
from keras.models import Sequential
from keras.layers import Conv1D, MaxPooling1D, Flatten, Dense
drive.mount('/content/drive')
# Cargar los datos desde los archivos CSV
data_corrientes =
np.loadtxt('/content/drive/MyDrive/DATA_IA/CORRIENTES_NORMAL_y_FALL
A.csv', delimiter=',')
data_salida =
np.loadtxt('/content/drive/MyDrive/DATA_IA/SALIDA_CLASES.csv',
delimiter=',')
# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test =
train_test_split(data_corrientes, data_salida, test_size=0.2,
random_state=42)
# Definir el modelo
model = Sequential()
# Primer bloque de capas convolucionales
model.add(Conv1D(64, kernel_size=3, activation='relu',
input_shape=(1000, 1)))
model.add(MaxPooling1D(pool_size=2))
# Segundo bloque de capas convolucionales
model.add(Conv1D(128, kernel_size=3, activation='relu'))
model.add(MaxPooling1D(pool_size=2))
```

```
# Tercer bloque de capas convolucionales
model.add(Conv1D(256, kernel_size=3, activation='relu'))
model.add(MaxPooling1D(pool_size=2))
# Cuarto bloque de capas convolucionales
model.add(Conv1D(512, kernel_size=3, activation='relu'))
model.add(MaxPooling1D(pool_size=2))
# Aplanar y agregar capas completamente conectadas
model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dense(512, activation='relu'))
# Capa de salida
model.add(Dense(1, activation='sigmoid'))
# Imprimir el resumen del modelo
model.summary()
# Compilar el modelo
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
# Entrenar el modelo
history=model.fit(X_train, y_train, epochs=50, batch_size=32,
validation_data=(X_test, y_test))
import tensorflow as tf
# Guardar el modelo entrenado
tf.saved_model.save(model,
'/content/drive/MyDrive/IA_ENTRENAMIENTO/MODELOCNN_2_IA_HIF')
```

Anexo 6. Codificación en Google Colab para el entrenamiento del algoritmo SVM

```
# Paso 1: Importar las bibliotecas necesarias
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.svm import SVC
import seaborn as sns
from sklearn.metrics import accuracy_score, classification_report
from google.colab import drive
from sklearn.decomposition import PCA
from sklearn.metrics import confusion_matrix,
classification_report, roc_curve, auc, accuracy_score
import joblib
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
drive.mount('/content/drive')
# Cargar los datos desde los archivos CSV
```

```
data_corrientes =
np.loadtxt('/content/drive/MyDrive/DATA_IA/CORRIENTES_NORMAL_y_FALL
A.csv', delimiter=',')
data_salida =
np.loadtxt('/content/drive/MyDrive/DATA_IA/SALIDA_CLASES.csv',
delimiter=',')
# Dividir el conjunto de datos en entrenamiento y prueba
X_train, X_test, y_train, y_test =
train_test_split(data_corrientes, data_salida, test_size=0.2,
random_state=42)
#Crear el modelo SVM
scaler = StandardScaler()
svm = SVC(kernel='linear', C=0.01, gamma='scale', probability=True,
verbose=True)
model = make_pipeline(scaler, svm)
# Entrenar el modelo
model.fit(X_train, y_train)
# Predecir en el conjunto de prueba
y_pred = model.predict(X_test)
# Calcular la precisión del modelo
accuracy = accuracy_score(y_test, y_pred)
print("Precisión del modelo:", accuracy)
# Generar la matriz de confusión
conf_matrix = confusion_matrix(y_test, y_pred)
# Visualizar la matriz de confusión
plt.figure(figsize=(10, 7))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Etiqueta Predicha')
plt.ylabel('Etiqueta Verdadera')
plt.title('Matriz de Confusión')
plt.show()
# Calcular las probabilidades de las predicciones
y_prob = model.predict_proba(X_test)[:, 1]
# Calcular la curva ROC
fpr, tpr, thresholds = roc_curve(y_test, y_prob)
roc_auc = auc(fpr, tpr)
# Visualizar la curva ROC
plt.figure(figsize=(10, 7))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve
(area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()
```

```
# Generar y mostrar el reporte de clasificación
report = classification_report(y_test, y_pred)
print("Reporte de Clasificación:")
print(report)
# Guardar el modelo entrenado en una carpeta específica en Google
Drive
folder_path = '/content/drive/MyDrive/IA_ENTRENAMIENTO'
model_path = f"{folder_path}/SVM1.pkl"
joblib.dump(model, model_path)
print(f"Modelo guardado como '{model path}'")
```

Anexo 7. Codificación para evaluar los algoritmos de inteligencia artificial propuestos

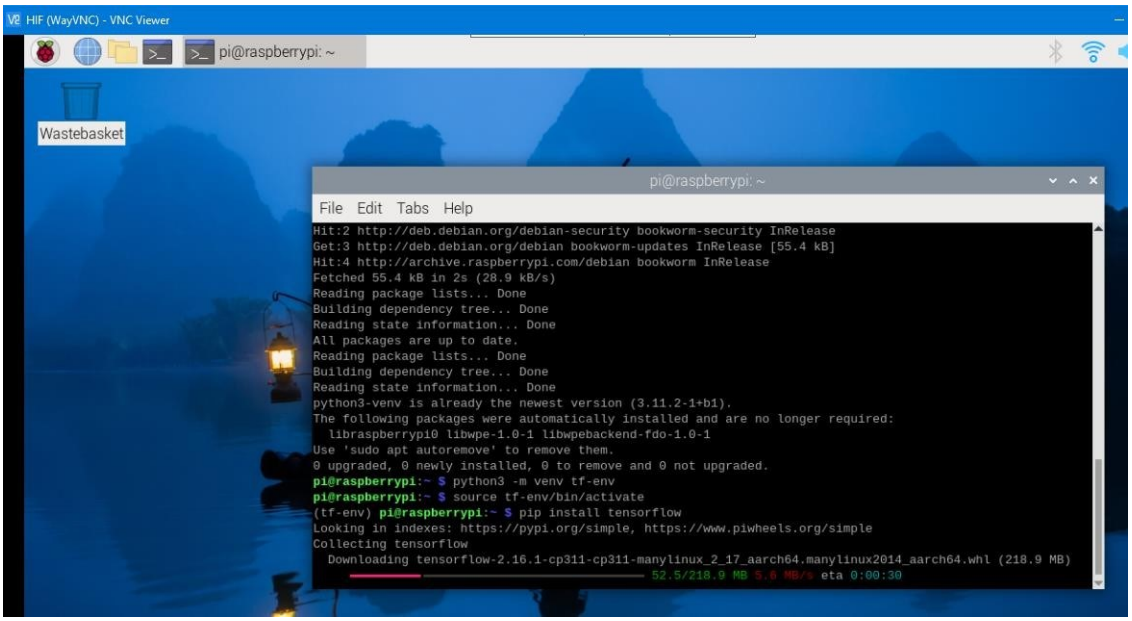
```
import tensorflow as tf
import numpy as np
# Ruta del directorio que contiene el modelo guardado
model_path =
'/content/drive/MyDrive/IA_ENTRENAMIENTO/MODELOMPL_2_IA_HIF'
# Cargar el modelo
model = tf.saved_model.load(model_path)
# Cargar los datos desde los archivos CSV
#data_corrientes_test =
np.loadtxt('/content/drive/MyDrive/DATA_IA/CORRIENTES_NORMAL_y_FALL
A_REAL_DATA.csv', delimiter=',')
data_corrientes_test =
np.loadtxt('/content/drive/MyDrive/DATA_IA/CORRIENTES_NORMAL_y_FALL
A_REAL2_DATA.csv', delimiter=',')
# Convertir las entradas a tensores tf.constant
input_data = tf.constant(data_corrientes_test, dtype=tf.float32)
# Realizar predicciones utilizando el modelo
predictions = model(input_data)
# Visualizar las predicciones
print("Predicciones:")
for i, pred in enumerate(predictions):
    print("Muestra %d: %s" % (i+1, pred))
#definir el conjunto de datos de la clase (salida real)
SALIDA_REAL =
np.loadtxt('/content/drive/MyDrive/DATA_IA/SALIDA_CLASES_NEW2.csv',
delimiter=',')
#realizar predicciones y comparar la predicción con la etiqueta
real
print("Predicciones vs. Salida Real:")
for i, (pred, true_label) in enumerate(zip(predictions,
SALIDA_REAL)):
    # Convertir la predicción a una etiqueta de clase (0 o 1)
```

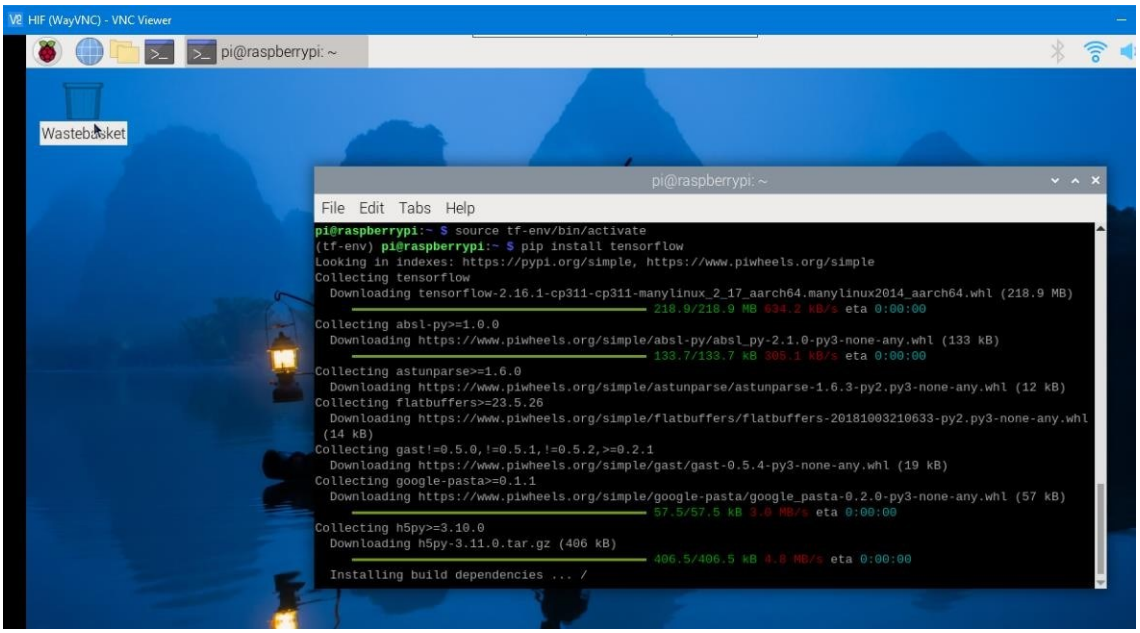
```
    predicted_label = 1 if pred > 0.5 else 0

    # Mostrar la salida predicha junto con la salida real
    print("Muestra %d - Predicha: %d, Real: %d" % (i+1,
predicted_label, true_label))
correct_predictions_0 = 0  # Inicializar contador de predicciones
correctas para la clase 0
correct_predictions_1 = 0  # Inicializar contador de predicciones
correctas para la clase 1
print("Predicciones vs. Salida Real:")
for i, (pred, true_label) in enumerate(zip(predictions,
SALIDA_REAL)):
    # Convertir la predicción a una etiqueta de clase (0 o 1)
    predicted_label = 1 if pred > 0.5 else 0
    # Mostrar la salida predicha junto con la salida real
    print("Muestra %d - Predicha: %d, Real: %d" % (i+1,
predicted_label, true_label))
    # Contar predicciones correctas para la clase 0 y para la clase
1
    if true_label == 0 and predicted_label == 0:
        correct_predictions_0 += 1
    elif true_label == 1 and predicted_label == 1:
        correct_predictions_1 += 1
# Imprimir el número de predicciones correctas para la clase 0 y
para la clase 1
print("Predicciones correctas para la clase 0:",
correct_predictions_0)
print("Predicciones correctas para la clase 1:",
correct_predictions_1)
# Contar el número de muestras de clase 0 y de clase 1 en
SALIDA_REAL
num_samples_class_0 = np.sum(SALIDA_REAL == 0)
num_samples_class_1 = np.sum(SALIDA_REAL == 1)
# Sumar las muestras de clase 0 y clase 1 para obtener el total
total_samples = num_samples_class_0 + num_samples_class_1
print("Número de muestras de clase 0:", num_samples_class_0)
print("Número de muestras de clase 1:", num_samples_class_1)
print("Número total de muestras:", total_samples)
# Calcular porcentaje de predicciones correctas e incorrectas para
la clase 0
accuracy_class_0 = (correct_predictions_0 / num_samples_class_0) *
100
error_rate_class_0 = 100 - accuracy_class_0
# Calcular porcentaje de predicciones correctas e incorrectas para
la clase 1
accuracy_class_1 = (correct_predictions_1 / num_samples_class_1) *
100
error_rate_class_1 = 100 - accuracy_class_1
# Imprimir los porcentajes
```

```
print("Porcentaje de predicciones correctas para la clase 0:",
accuracy_class_0, "%")
print("Porcentaje de predicciones incorrectas para la clase 0:",
error_rate_class_0, "%")
print("Porcentaje de predicciones correctas para la clase 1:",
accuracy_class_1, "%")
print("Porcentaje de predicciones incorrectas para la clase 1:",
error_rate_class_1,"%")
```

Anexo 8. Instalación del framework tensorflow en el sistema embebido





Anexo 9. Script implementado en el sistema embebido

```
import tensorflow as tf
import numpy as np
import time
# Ruta del directorio que contiene el modelo guardado
model_path = '/home/pi/tf-env/IA/MODELOMPL_2_IA_HIF'
# Cargar el modelo
model = tf.saved_model.load(model_path)
# Cargar los datos desde los archivos CSV
data_corrientes_test = np.loadtxt('/home/pi/tf-env/IA/DATA_REAL1_CLASE0.csv',
delimiter=',')
salida_real = np.loadtxt('/home/pi/tf-env/IA/SALIDA_REAL1_CLASE0.csv',
delimiter=',')
# Convertir las entradas a tensores tf.constant
input_data = tf.constant(data_corrientes_test, dtype=tf.float32)
# Inicializar lista para tiempos de procesamiento y etiquetas predichas
processing_times = []
predicted_labels = []
# Realizar predicciones utilizando el modelo y medir el tiempo de procesamiento
print("Predicciones vs. Salida Real:")
for i, (sample, true_label) in enumerate(zip(input_data, salida_real)):
    start_time = time.time()
    prediction = model(tf.expand_dims(sample, axis=0))  # Expandir dimensiones para
que sea compatible con el modelo
    end_time = time.time()
    processing_time = end_time - start_time
    processing_times.append(processing_time)
    # Convertir la predicciOn a una etiqueta de clase (0 o 1)
    predicted_label = 1 if prediction > 0.5 else 0
    predicted_labels.append(predicted_label)
    # Mostrar la salida predicha junto con la salida real y el tiempo de procesamiento
    print("Muestra %d - Predicha: %d, Real: %d (Tiempo de procesamiento: %.6f
segundos)" % (i + 1, predicted_label, true_label, processing_time))
    # Cada 30 muestras, calcular la clase mayoritaria y mostrar el mensaje adecuado
    if (i + 1) % 30 == 0:
        count_0 = predicted_labels[-30:].count(0)
        count_1 = predicted_labels[-30:].count(1)
        if count_0 > count_1:
            print("FLUJO NOMINAL DE CORRIENTE ELECTRICA")
        else:
            print("ALARMA FALLA DE ALTA IMPEDANCIA")
# Calcular y mostrar el tiempo promedio de procesamiento
average_time = sum(processing_times) / len(processing_times)
print("\nTiempo promedio de procesamiento por muestra: %.6f segundos" %
average_time)
```