ISBN: 978-628-96613-0-9. ISSN: 2414-6390. Digital Object Identifier: https://dx.doi.org/10.18687/LEIRD2024.1.1.802

Comparison between the prediction of an artificial intelligence and the calculation of the mathematical formula in the operation time of a relay with overcurrent protection 51 with very inverse ANSI curve

Darío Gómez Esteba, Ms.[1], César Castillo Cáceres, Dr.[2], Lucy Giannina Ticona Huilca, Dr.[3], y Deidamia Giovanna Chani Ollachica, Dr.[4],

[1]*Universidad Nacional de San Agustín, Perú, darioge23@gmail.com*
[2]*Universidad Católica de Santa María, Perú, ccastill@ucsm.edu.pe*
[3]*Universidad Nacional de San Agustín, Perú, lticonah@unsa.edu.pe*
[4]*Universidad Nacional de San Agustín, Perú, dchani@unsa.edu.pe*

Abstract– ***This work focuses on making a comparison between the prediction of an artificial intelligence and a mathematical formula that calculates the operation time of an overcurrent protection relay with a very inverse ANSI curve. To make this comparison, first, a neural network was built to learn to predict the operation time of an overcurrent protection relay with a very inverse ANSI curve. Then, a web interface was developed where the results of the neural network's prediction and the results of the mathematical calculation using the aforementioned formula can be visualized.***

To build the neural network, the Google Colab interface was used with the Python programming language, and to create the web interface, HTML, CSS, and JavaScript were employed. With the neural network and web interface developed, a comparison is carried out between the neural network's prediction and a mathematical formula that calculates the operation time of an overcurrent protection relay with a very inverse ANSI curve using three examples, along with discussions, observations, and conclusions.

Keywords-- ***Artificial Intelligence, Neural Network, Over current Protection 51, Curve ANSI very invers.***

Comparación entre predicción IA y el cálculo del tiempo de operación del relé con protección de sobrecorriente 51 curva ANSI inversa

Darío Gómez Esteba, Ms.[1], César Castillo Cáceres, Dr.[2], Lucy Giannina Ticona Huilca, Dr.[3], y Deidamia Giovanna Chani Ollachica, Dr.[4],

[1]*Universidad Nacional de San Agustín, Perú, darioge23@gmail.com*
[2]*Universidad Católica de Santa María, Perú, ccastill@ucsm.edu.pe*
[3]*Universidad Nacional de San Agustín, Perú, lticonah@unsa.edu.pe*
[4]*Universidad Nacional de San Agustín, Perú, dchani@unsa.edu.pe*

Resumen– ***Este trabajo se enfoca en realizar una comparación entre la predicción de una inteligencia artificial y una fórmula matemática que calcula el tiempo de operación de un relé de protección de sobre corriente con curva ANSI very inverse.***

Para hacer esta comparación primero se construyó una red neuronal que aprenda a predecir el tiempo de operación de un relé de protección de sobre corriente con curva ANSI very inverse; luego se desarrolla una interfaz web donde se puede visualizar los resultados de la predicción de la red neuronal y los resultados del cálculo matemático con la formula mencionada anteriormente.

Para construir la red neuronal se utilizó la interfaz Google Colab con lenguaje de programación Python; y para construir la Interfaz web se utilizó HTML, CSS y Javascript.

Con la red neuronal y la interfaz web desarrollados, se realiza la comparación entre la predicción de la red neuronal y una fórmula matemática que calcula el tiempo de operación de un relé de protección de sobre corriente con curva ANSI very inverse.

Palabras clave– Inteligencia artificial, Red neuronal, Protección Contra Sobrecorriente 51, Curva ANSI muy inversa.

I. Introducción

En un mundo donde la inteligencia artificial avanza a pasos agigantados, este trabajo busca realizar una comparación entre la predicción de una inteligencia artificial y una fórmula matemática que calcula el tiempo de operación de un relé de protección de sobre corriente con curva ANSI very inverse. También se desarrolló una interfaz web que realice la predicción y el cálculo del tiempo de operación de un relé con protección de sobre corriente 51 con curva ANSI very inverse; la Interfaz web necesita como valor de entrada a la corriente de sobrecarga.

Primero se creó un algoritmo de aprendizaje automático en la plataforma Google Colab con lenguaje de programación Python que realiza la predicción del tiempo de operación de un relé de protección de sobre corriente 51 con curva ANSI very inverse.

Con nuestro modelo entrenado lo llevaremos a una Interfaz web para que pueda ser comparado con el cálculo de la fórmula matemática usando lenguaje de programación JavaScript con apoyo de TensorFlow.js, HTML, CSS.

Esta tesis también busca innovar y apoyar en futuras investigaciones para crear un asistente virtual hecho con inteligencia artificial; que pueda realizar cualquier tipo de predicciones matemáticas y brindar cualquier tipo de información en el ámbito de la Ingeniería Eléctrica.

A. Corriente de sobrecarga

Es la corriente que supera la capacidad nominal diseñada de un equipo, dispositivo o circuito eléctrico el cual causa daños y disminución de su vida útil.

B. Protección contra sobrecorriente 51

Conocida también como protección de sobre corriente a tiempo inverso; es un tipo de protección eléctrica que se usa para proteger equipos eléctricos y circuitos eléctricos de corrientes de sobrecarga, de acuerdo al estándar IEEE Std C37.112-1996. El tiempo de actuación del relé es variable según la corriente detectada por el relé. [2]

C. Curva tiempo inverso

C.1. Tipos de curva tiempo inverso ANSI

(U1) MODERADAMENTE INVERSO.

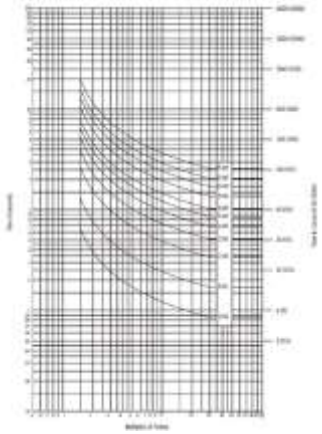

Fig. 1 Curva ANSI Moderadamente Inverso [1]

(U2) INVERSO.

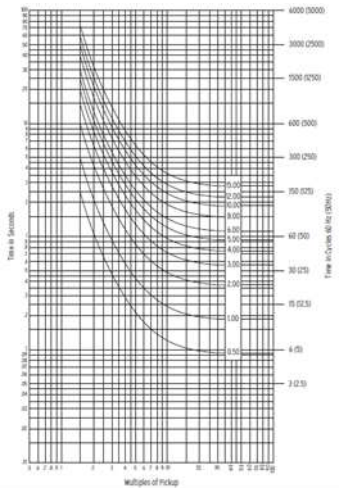

Fig. 2 Curva ANSI Inverso [1]

(U3) MUY INVERSO.

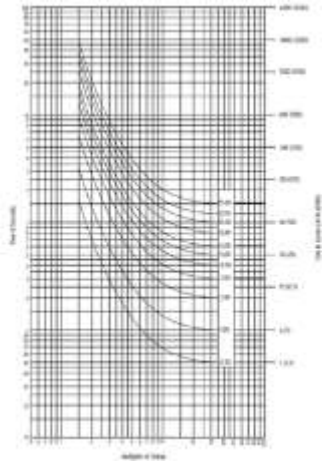

Fig. 3 Curva ANSI Muy Inverso [1]

C.2. Tipos de curva tiempo inverso IEC

(C1) INVERSO ESTÁNDAR

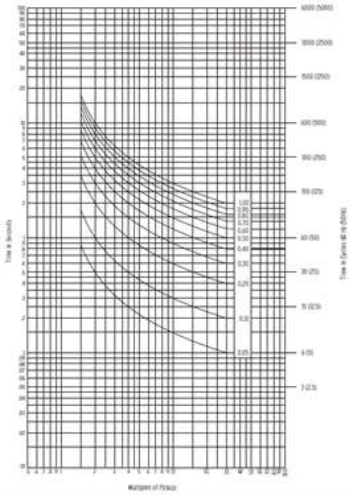

Fig. 4 Curva IEC Inverso Estándar [1]

D. Corriente pick up

Es la corriente inicial que empezará a reconocer el relé según la curva tiempo-inverso; la curva tiempo inverso se desplaza horizontalmente cuando cambiamos los valores de la corriente pick up. [3]

E. Dial

Es un valor adimensional usado para modificar la curva tiempo-inverso de forma vertical, es muy usado para coordinar protecciones.

F. Múltiplo de pick up

El múltiplo de Pick Up es la relación entre la corriente de sobrecarga que detecto el relé y la corriente pick up. Se considera frecuentemente en los relés como el eje (x) de la curva tiempo inverso. [3]

$$M = \frac{Isobrecarga}{Ipickup\,(51)} \quad (1)$$

G. Tiempo de operación

Es el tiempo de operación diseñado para que el relé realice un trip ante una corriente de sobrecarga. Se ubica en el eje (y) de la curva tiempo inverso.

TIEMPO DE OPERACIÓN (U1)

$$tp = TD * (0.0226 + \frac{0.0104}{M^{0.02} - 1}) \quad (2)$$

TIEMPO DE OPERACIÓN (U2)

$$tp = TD * (0.180 + \frac{5.95}{M^{2} - 1}) \quad (3)$$

TIEMPO DE OPERACIÓN (U3)

$$tp = TD * (0.0963 + \frac{3.88}{M^{2} - 1}) \quad (4)$$

TIEMPO DE OPERACIÓN (U4)

$$tp = TD * (0.0352 + \frac{5.67}{M^{2} - 1} \quad (5)$$

TIEMPO DE OPERACIÓN (U5)

$$tp = TD * (0.00262 + \frac{0.00342}{M^{0.02} - 1}) \quad (6)$$

TIEMPO DE OPERACIÓN (C1)

$$tp = TD * (\frac{0.14}{M^{0.02} - 1}) \quad (7)$$

TIEMPO DE OPERACIÓN (C2)

$$tp = TD * (\frac{13.5}{M-1}) \quad (8)$$

TIEMPO DE OPERACIÓN (C3)

$$tp = TD * (\frac{80}{M^2-1}) \quad (9)$$

TIEMPO DE OPERACIÓN (C4)

$$tp = TD * (\frac{120}{M-1}) \quad (10)$$

TIEMPO DE OPERACIÓN (C5)

$$tp = TD * (\frac{0.05}{M^{0.04}-1}) \quad (11)$$

H. Aprendizaje profundo (Deep Learning)

Es una de las áreas más activas en la inteligencia artificial el cual se están dando innovaciones importantes en todo el mundo, el Deep Learning se inspira en el cerebro humano, en vez de programar un algoritmo que solo realiza tareas señaladas; se entrena un modelo de aprendizaje para que pueda realizar predicciones. [4]

I. Redes neuronales artificiales

Las redes neuronales son modelos matemáticos que están hecho de neuronas interconectadas, a estas neuronas se les entrena y aprenden patrones para que puedan procesar cualquier información.

J. Capas ocultas

Las redes neuronales tienen capas ocultas para que puedan aprender predicciones más complejas.

K. Entrenamiento

Para que las redes neuronales realicen predicciones; primero deben aprender, es por eso que se tiene que entrenar primero. Se asigna las variables y los resultados respectivos para que tenga un aprendizaje profundo; una vez entrenado las redes neuronales estarán listos para realizar la predicción deseada.

L. Funciones de activación

Dentro de cada neurona existe la función de activación que activa o desactiva el aprendizaje por neurona, cada neurona realiza una revisión si el aprendizaje obtenido es efectivo, si no lo es; no deja pasar ese aprendizaje a la siguiente neurona. Logrando así predicciones más complejas.

M. Función de activación RELU

Es una de las funciones de activación más utilizadas en la predicción de redes neuronales, su forma matemática es bastante simple; ello permite que las iteraciones no sean muy complejas y tomen menor tiempo.

N. Época

Es una iteración con todos los ejemplos de muestra, la época se pude dividir para calcular la estimación, para dividirlo se usa el comando batch size.

O. Batch Size

Se utiliza este comando para dividir la época y que la iteración se realice en paralelo.

P. Optimizador con algoritmo ADAM

Optimiza el aprendizaje de la red neuronal para que aprenda y no desaprenda en el menor número de iteraciones. Es una de las herramientas más utilizadas en la optimización de redes neuronales. Es la combinación del método momentum y rmsprop, que son usadas para que la predicción sea más efectiva.

Se regula la optimización con el learning rate, que no debe ser ni muy grande por que obtiene valores muy grandes al momento de aprender; ni muy pequeño por que será difícil optimizar la predicción. Un valor aproximado del learning rate es el 0.01; este valor puede variar de acuerdo a la dificultad de la predicción.

Q. Transferencia de aprendizaje

Muchas veces el entrenamiento de una red neuronal se guarda en un archivo ".json" y ".bin" y se transfiere con JavaScript para darle aplicación en cualquier Interfaz o lenguaje de programación diferente. [5], [6].

R. Python

Es un lenguaje de programación muy conocido y muy utilizado, tiene facilidad de escritura y lectura; es usado muchas aplicaciones desde desarrollo web hasta aprendizaje automático. El código de Python se interpreta línea por línea, lo que lo hace más accesible, se puede escribir en Python y puede ser usado en muchas plataformas como Windows, Linux sin tener que realizar muchos cambios. Se puede encontrar muchos algoritmos ya creados por varios usuarios. Python tiene muchas librerías de gran utilidad como Numpy, pandas, matplotlib, Flask, Tensorflow, Pytorch, etc.

Tiene una sintaxis clara, además este lenguaje de programación es de código abierto y es gratuito.[7]

Fig. 5 Icono Python [7]

S. Javascript

Es un lenguaje de programación muy usado en aplicaciones web, se puede ejecutar en tiempo real en un navegador web.Con javascript se puede crear, modificar aplicaciones web de acuerdo a los gustos del usuario, es compatible con los principales navegadores web; su lenguaje de programación se asemeja a javascript y C++; cuenta con una variedad amplia de bibliotecas y frameworks. También es usado para realizar predicciones de modelos ya entrenados, el cual solo necesita el archivo json y bin. [8]

Fig. 6 Icono JavaScript [8]

T. HTML (Hyper Text Markup Language)

Es muy usado para definir la estructura y presentación de un navegador web, a continuación, se presenta la estructura básica:

<html> define el inicio del documento

<head> es el encabezado y contiene títulos y enlaces a CSS y Script.

<body> contiene el título, imágenes formas, cuadros que se pueden visualizar en la Interfaz web.

También tiene elementos que pueden interactuar con el navegador web con botones, inputs, salidas, etc. Todo depende de las necesidades del usuario. Es compatible con los navegadores modernos; y su uso va acompañado de CSS donde puedes definir la presentación, también de JavaScript que define la interactividad de la Interfaz web. [9]

Fig. 7 Icono HMTL [9]

U. Css (Cascading Style Sheets)

Es un lenguaje donde puedes diseñar la presentación de una Interfaz web. Está vinculado al HTML. Puedes cambiar el tamaño del texto, el color del texto, el tipo de letra, crear formas geométricas, modificar el tamaño de las imágenes, etc. Es muy importante para diseñar una Interfaz web de acuerdo a los gustos del usuario. [10]

Fig. 8 Icono CSS [10]

V. Google Colab

Es una plataforma que permite que varios usuarios puedan interactuar con lenguaje de programación Python, y con todas sus librerías. También permite el acceso gratuito a Unidades de procesamiento Grafico (GPUs) y Unidades de Procesamiento Tensorial (TPUs), el cual es necesario para realizar cálculos de aprendizaje profundo. [11]

Fig. 9 Icono Google Colab [11]

II. MATERIALES Y MÉTODOS

Para desarrollar la Interfaz web que realice una comparación entre la predicción de una inteligencia artificial y una fórmula matemática que calcula el tiempo de operación de un relé de protección de sobre corriente con curva ANSI very inverse, primero se tiene que desarrollar con lenguaje de programación Python usando bibliotecas como numpy, matplotlib, tensorflow y pandas en la plataforma Google colab.

El algoritmo tendrá la función de recopilar datos de entrada necesarios para predecir el tiempo de operación de un relé de protección de sobre corriente 51 con curva ANSI very inversa; estos datos serán procesados por una red neuronal conformada por 6 capas ocultas con más de 1 neurona cada una; el cual serán entrenados hasta que la predicción tenga errores mínimos. Una vez realizado la predicción con errores mínimos se guarda todo el proceso en dos archivos ".json" y ".bin".

Como se necesita realizar la predicción en una Interfaz web, se tiene que exportar el archivo ". json" en una plataforma donde te permita crear una Interfaz web. Para esto se usa HTML, CSS, JavaScript, Python. HTML es para crear la estructura de la Interfaz web, CSS es para diseñar la forma de la Interfaz web, JavaScript es para importar el archivo ".json" y poder realizar la predicción en la Interfaz web; asimismo se desarrolla en la interfaz web el cálculo con fórmula matemática desarrollada en javascript y así se pueda observar la diferencia entre la predicción de una red neuronal y la fórmula matemática que calcula el tiempo de operación de un relé de protección de sobre corriente con curva ANSI very inverse.

A. Modelo matemático

Para poder predecir el tiempo de operación de un relé con protección 51 con curva ANSI very inverse, se usa como referencia la siguiente formula:

$$tp = TD * (A + \frac{B}{(\frac{Ie}{Ipu})^2 - 1}) \quad (11)$$

tp= Tiempo de operación del trip del relé
Ie= Corriente de entrada o también de sobrecarga
Ipu= Corriente pick up de protección 51
TD= Dial
A=0.0963
B=3.88

Los valores como tp, TD, A, B, Ie, Ipu, C, son ingresados en una base de datos de Excel para que la red neuronal pueda lecturar y así poder realizar una predicción con errores mínimos.

B. Base de datos en Excel para entrenar la red neuronal "ANSI-VI.xlsx"

Para poder entrenar la red neuronal se necesita una base de datos de los parámetros de la fórmula ANSI very inverse, que está almacenado en un archivo Excel con nombre (ANSI-VI.xlsx). La base de datos tiene 20000 ejemplos para realizar el entrenamiento de la red neuronal. Está conformada por 6 columnas (Ie, Ipu, TD, A, B, tp) y 20000 filas.

Ie= Corriente de entrada o de sobrecarga de la formula ANSI very inverse
Ipu= Corriente pick up protección 51 de la formula ANSI very inverse
TD= Dial de la formula ANSI very inverse
A= 0.0963 Parámetro de la formula ANSI very inverse
B= 3.88 Parámetro de la formula IEC very inverse
tp= tiempo de operación de la formula ANSI very inverse

	A	B	C	D	E	F
1	Ie	Ipu	TD	A	B	tp
2	150	120	0,500	0,0963	3,880	3,49704
3	151	120	0,500	0,0963	3,880	3,37347
4	152	120	0,500	0,0963	3,880	3,25771
5	153	120	0,500	0,0963	3,880	3,14905
6	154	120	0,500	0,0963	3,880	3,04686
7	155	120	0,500	0,0963	3,880	2,95059

Fig. 10 Datos en archivo "ANSI-VI.xlsx

"

C. Red neuronal creada con PYTHON usando la plataforma Google Colab.

Para poder realizar la predicción se creó una red neuronal con lenguaje de programación Python en la plataforma Google Colab. Primero ingresamos a Google Colab y declaramos que de Google colab se puede cargar y descargar archivos.

```
from google.colab import files
```

Se declara una variable load que activa la función cargar archivos del módulo file que ofrece Google Colab.

```
load=files.upload()
```

Se importa las bibliotecas:

-Pandas: Es una biblioteca de Python que nos ofrece interactuar con base de datos.

-Numpy: Es una biblioteca que nos proporciona interactuar con matrices y arreglos multidimensionales y matrices, también permite realizar operaciones matemáticas y científicas.

-Tensorflow: Es una biblioteca de código abierto creado por Google para crear y entrenar redes neuronales.

-Matplotlib: Es una biblioteca de Python que permite crear gráficos y visualizar datos.

```
import pandas as pd
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
```

Se declara en una variable "datos" toda la base de datos almacenada en el archivo Excel usando la biblioteca pandas.

```
datos=pd.read_excel(load["ANSI-VI.xlsx"])
```

Se extrae de la base de datos las variables Ie, Ipu, TD, A, B, tp y se ordena en matrices de 1x20000.

```
Ie= np.array(datos['Ie']).reshape((-1,1))
Ipu= np.array(datos['Ipu']).reshape((-1,1))
TD= np.array(datos['TD']).reshape((-1,1))
A= np.array(datos['A']).reshape((-1,1))
B= np.array(datos['B']).reshape((-1,1))
tp= np.array(datos['tp']).reshape((-1,1))
```

Se une las matrices Ie, Ipu, TD, A, B en una sola matriz llamada x.

```
x= np.hstack((Ie, Ipu, TD, A, B))
```

Se crea 1 capa de entrada con 5 input, 6 capas ocultas y una capa de salida con 1 output; la capa oculta 1 tiene 400 neuronas, la capa oculta 2 tiene 200 neuronas, la capa oculta 3 tiene 100 neuronas, la capa oculta 4 tiene 50 neuronas, la capa oculta 5 tiene 10 neuronas y la capa oculta 6 tiene 3 neuronas. Para crear esta estructura neuronal se utiliza la biblioteca de código abierto keras, tensorflow; esta red neuronal se está programando de forma densa (que hay una comunicación entre todas las neuronas para un aprendizaje eficaz). Se usa el optimizador Adam, que permite que las neuronas aprendan en vez de desaprender. También se ejecuta el entrenamiento del modelo de red neuronal con el comando fit; se ingresa como variable de entrada la matriz x y como resultado la matriz tp, todo esto se almacena en una variable llamada "historial". Para el optimizador Adam se utiliza un valor de 0.0001890001. Se programa 40000 épocas y 20000 de batch size. El tiempo que demora el entrenamiento de la red neuronal con los parámetros mencionados es de 15 horas.

```
oculta1=tf.keras.layers.Dense(units=400,
input_shape=(5,), activation='relu')
oculta2=tf.keras.layers.Dense(units=200,activation='relu')
oculta3=tf.keras.layers.Dense(units=100,activation='relu')
oculta4=tf.keras.layers.Dense(units=50,activation='relu')
oculta5=tf.keras.layers.Dense(units=10,activation='relu')
oculta6=tf.keras.layers.Dense(units=3,activation='relu')
salida=tf.keras.layers.Dense(units=1,activation='relu')
modelo=tf.keras.Sequential([oculta1, oculta2, oculta3,
oculta4, oculta5, oculta6, salida])
modelo.compile(

optimizer=tf.keras.optimizers.Adam(learning_rate=0.000189
0001),
    loss='mean_squared_error')
# Entrenar el modelo
print("Comenzando entrenamiento...")
historial= modelo.fit(x, tp, epochs=20000,
batch_size=20000)
print("Modelo entrenado!")
```

Una vez entrenado la red neuronal con época=40000, y batch size= 20000.

Se logró un error de 0.0848.

```
1/1 [==============================] - 0s
336ms/step - loss: 0.2199
Epoch 19997/20000
1/1 [==============================] - 0s
327ms/step - loss: 0.0985
Epoch 19998/20000
1/1 [==============================] - 0s
331ms/step - loss: 0.0431
Epoch 19999/20000
1/1 [==============================] - 0s
335ms/step - loss: 0.0436
Epoch 20000/20000
1/1 [==============================] - 0s
375ms/step - loss: 0.0848
Modelo entrenado!
```

En la siguiente se pone a prueba el modelo entrenado; también se utiliza la biblioteca de matplotlib para crear un gráfico, en el eje "x" esta la época y en el eje "y" esta la magnitud de perdida, el comando print es para mostrar los valores que se quiere.

```
plt.xlabel("# Epoca")
plt.ylabel("Magnitud de pérdida")
plt.plot(historial.history["loss"])
resultado = modelo.predict([[150, 120, 0.5, 0.0963,
3.88]])
print("Probando nos da un resultado de:")
print("El resultado es " + str(resultado))
```

Del anterior comando podemos observar la predicción y la gráfica de la magnitud de perdida con el número de épocas.

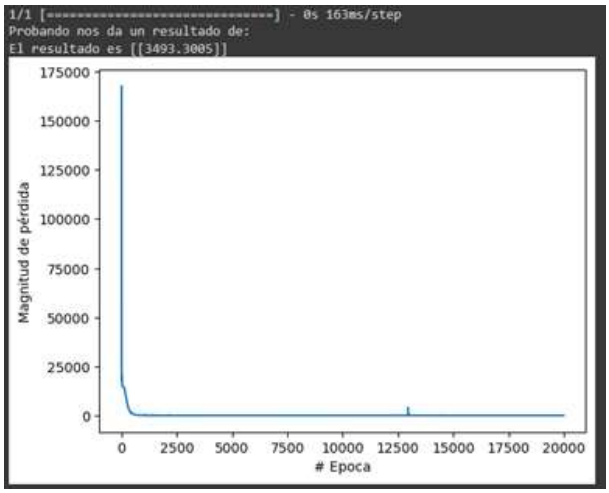

Fig. 11 Grafica Magnitud de Perdida VS Epoca

El siguiente comando es para exportar la red neuronal en un archivo h5 con nombre "TESIS.h5".

```
#Exportar el modelo en formato h5
modelo.save('TESIS.h5')
```

Para convertir a tensorflow.js primero instalamos la librería.

```
#Para convertir a tensorflow.js,primero se instala la
libreria
!pip install tensorflowjs
```

Se crea una carpeta llamada "carpeta_salida".

```
#Crear carpeta donde se colocarán los archivos
resultantes
!mkdir carpeta_salida
```

Con el siguiente comando se convierte el archivo "TESIS.h5" a dos archivos; llamados "model.json" y "group1-shardof1.bin"; el archivo "model.json" se usa para que pueda hacer la predicción en una Interfaz web.

```
#Realizar la exportacion a la carpeta de salida
!tensorflowjs_converter --input_format keras TESIS.h5
carpeta_salida
```

D. Creación de la Interfaz Web

Los archivos HTML "tesis.html", CSS "style.css", JavaScript "script.js", Excel "ANSI-VI.xlsx", JSON "model.json" y las imágenes que se utiliza en la Interfaz web se almacenan en una carpeta llamada "INTERFAZ WEB".

Fig. 12 Carpeta "INTERFAZ WEB"

A continuación, se observa los archivos guardados en la carpeta "INTERFAZ WEB".

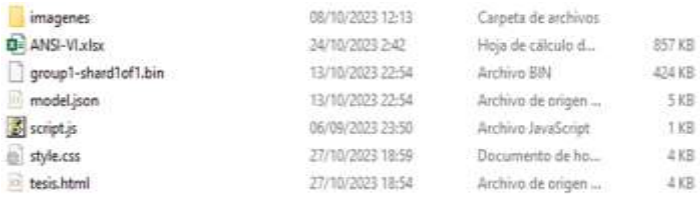

Fig. 13 Archivos en la carpeta "INTERFAZ WEB"

En la carpeta imágenes se puede mostrar todas las imágenes que se usó para la Interfaz Web.

Fig. 14 Imágenes en la carpeta "INTERFAZ WEB"

E. Archivo HTML "TESIS.HTML"

Para crear una Interfaz web se necesita HTML donde se pueda modificar la estructura, que se mostrará a continuación:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">

<div class="content">

   <div data-content id="ansi">
      <table class="tabla">
         <tr>
         </tr>
      </table>
         <h1 class="texto-Ie">I Sobrecarga</h1>
         <h1 class="texto-IA">IA</h1>
         <h1 class="texto-formula">FORMULA</h1>
         <h1 class="texto-Ipu-51">Ipickup (51)</h1>

            var prediccion = modelo.predict(tensor).dataSync();
            prediccion = Number(prediccion).toFixed(2);
            document.getElementById("itp").value = prediccion;

            let formula= Number((0.5*(0.0963+(3.88/(((Ie/120)*(Ie/120))-1)))*(1000)).toFixed(2));
            document.getElementById("itp1").value = formula;
            }
         </script>
</div></div>
```

Para un mayor entendimiento se mostrará por partes el algoritmo general HTML. Primero aquí se muestra la estructura inicial del HTML, el nombre que representa a la Interfaz web como "TESIS DARIO".

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>TESIS DARIO</title>
   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"rel="stylesheet"integrity="sha384EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">
```

Ahora se hace un puente con el archivo CSS "style.css" para ver los detalles de color, tamaño de letra, tamaño de las imágenes…etc. También se hace un puente con el archivo JavaScript "script.js".

```
<link rel="stylesheet"href="style.css">
   <script src="script.js" defer></script>
```

En el siguiente algoritmo se está importando dos imágenes en la ventana principal que son el logotipo de la UNSA y una imagen de una cabeza robótica.

```
<img src="imagenes/LOGO UNSA.png" class="imagen-unsa">
   <img src="imagenes/IA-ELECTRICAL.png" class="imagen-IA"></div>
```

A continuación, se está creando una tabla que va a contener información como la corriente pick up de la protección 51, dial de la protección 51.

```
<div class="content">
<div data-content id="ansi">
      <table class="tabla">
         <tr>
            <th></th>
            <th>I pickup(51)</th>
```

Se muestra el texto que está agregado a la Interfaz web.

```
<h1 class="texto-Ie">I Sobrecarga</h1>
         <h1 class="texto-IA">IA</h1>
         <h1 class="texto-formula">FORMULA</h1>
         <h1 class="texto-Ipu-51">Ipickup (51)</h1>
         <h1 class="texto-tp-51">tp(Tiempo de Operacion 51)</h1>
         <h1 class="texto-tp-51-1">tp(Tiempo de Operacion 51)</h1>
         <h1 class="texto-Ipu-50">Ipickup (50)</h1>
         <h1 class="texto-tp-50">Tiempo de Operacion (50)</h1>
```

Se muestra la entrada y la salida que tiene la Interfaz web, la entrada es la corriente de sobrecarga y la salida es el tiempo de operación del rele con protección 51.

```
<form>
<input id="iIe" type="number" name="Ie" onkeyup="calculartp();" class="ie">
<input id="itp" type="number" name="tp" readonly="" class="tp">
<input id="itp1" type="number" name="tp1" readonly="" class="tp1">
</form>
```

Se muestra las imágenes que se agregan en la ventana ANSI.

```
<img src="imagenes/CURVA-TIEMPO-CORRIENTE1.png" class="imagen-curva-tiempo-corriente">
```

Se muestra un lenguaje en java script para cargar el archivo "model.json". Luego se guarda en una matriz con las 5 entradas que necesita la predicción; las entradas son corriente de sobrecarga, corriente pick up de la protección 51 que tiene un valor de 120, el dial que es 0.5, parámetro 0.0963 y 3.88. Luego de realizar la predicción muestra en la Interfaz web el resultado que se desea obtener que en este caso es el tiempo de operación de un rele con protección 51.

```
<scriptsrc="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"integrity="sha384MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@2.0.0/dist/tf.min.js"></script>
<script type="text/javascript">
var modelo = null;
//Cargar modelo
(async () => {
console.log("Cargando modelo...");
modelo = await tf.loadLayersModel("model.json");
console.log("Modelo cargado...");
})();
function calculartp() {
var Ie = parseFloat(document.getElementById("iIe").value);
var matriz = new Array(Ie, 120, 0.5, 0.0963, 3.88);
var tensor = tf.tensor2d([matriz]);
var prediccion = modelo.predict(tensor).dataSync();
prediccion = Number(prediccion).toFixed(2);
document.getElementById("itp").value = prediccion;
```

Luego se escribe la fórmula matemática para que pueda ser comparado la predicción de la red neuronal:

```
let formula= Number((0.5*(0.0963+(3.88/(((Ie/120)*(Ie/120))1)))*(1000)).toFixed(2));
document.getElementById("itp1").value = formula;
</script>
</div>
</div>
</body>
</html>
```

F. Archivo CSS "STYLE.CSS"

Se muestra el archivo CSS "style.css" que está vinculado al archivo "tesis.html" donde se modifica el tamaño de letra, color de letra, estilo de letra, posición de letra, tamaño de imagen, posición de imagen, características de tabla, forma de input y output.

```
*{
margin: 0;
padding: 0;
box-sizing: border-box;
font-family: sans-serif;
}
body{
padding: 5%;
}
.menu{
display:flex;
}
.tp1{
position: absolute;
left: 450px;
top: 400px;
z-index: 3;
width: 110px;
height: 30px;
border-radius: 20px;

border: 2px solid;
color: rgb(3, 3, 3);
font-weight: bold;
text-align: center;

.table, th, td{

border: 1px solid black;
border-collapse: collapse;
text-align: center;
}
th, td{
padding: 2px;
}
```

G. Archivo Javascript "SCRIPT.JS"

Se muestra el archivo Java Script "script.js"; para interactuar con las ventanas en la Interfaz web; haciendo click en la palabra ANSI.

```
const targets = document.querySelectorAll('[data-target]')
const content = document.querySelectorAll('[data-content]')
targets.forEach(target =>{
  target.addEventListener('click',()=>{
    content.forEach(c =>{
      c.classList.remove('active')
    })
  const t = document.querySelector(target.dataset.target)
  t.classList.add('active')
})
})
```

H. Interfaz Web

De esta manera es como se puede visualizar la Interfaz web:

Fig. 15 "Interfaz Web"

Se hace click en la palabra ANSI para poder acceder a la interfaz web desarrollada; el cual está lista para ingresar el valor de la corriente de sobrecarga deseada para que calcule el tiempo de operación de un relé de protección de sobre corriente 51 con curva ANSI very inverse.

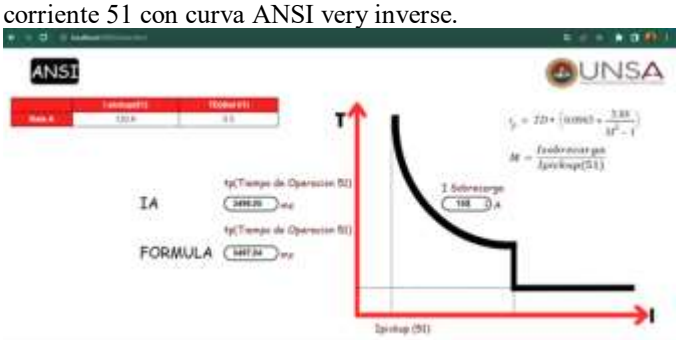

Fig. 16 "Interfaz WEB-ANSI"

III. RESULTADO Y DISCUSIÓN

A. Análisis e interpretación de los resultados

A.1. Calibración de la red neuronal

Para encontrar la predicción deseada y que la red neuronal tenga un aprendizaje con errores mínimos es necesario calibrar los siguientes parámetros:

- *Cantidad de ejemplos de aprendizaje*

Es la cantidad de parámetros ingresados en nuestro Excel "ANSI-VI.xlsx" como (Ie, Ipu, A, B, tp).

	Ie	Ipu	TD	A	B	tp
1	150	120	0,500	0,09630	3,880	3497,038889

Fig. 17 Ejemplos de Aprendizaje

- *Número de ejemplos de aprendizaje*

En nuestro caso se eligió 20000 ejemplos:

	Ie	Ipu	TD	A	B	tp
19998	20147	120	0,500	0,09630	3,880	48,218827
19999	20148	120	0,500	0,09630	3,880	48,21882017
20000	20149	120	0,500	0,09630	3,880	48,21881334

Fig. 18 Número de Ejemplos de Aprendizaje

- *Número de épocas*

Es cuando la red neuronal realiza el aprendizaje del número de ejemplos de que estan registrados en el Excel "ANSI-VI.xlsx".

- *Batch Size*

El batch size permite separar los ejemplos de entrada y realizar el aprendizaje de la red neuronal en paralelo.

- *Número de capas*

El número de capas conforman a la red neuronal.

- *Número de neuronas por capa*

Por cada capa existe una cantidad de neuronas.

- *Learning Rate del optimizador ADAM*

El learning rate permite que cada neurona aprenda y no desaprenda al realizar el aprendizaje. Se muestra 3 casos de estudio, donde se cambia el valor de la corriente de sobrecarga para evaluar y analizar la diferencia entre la predicción de la red neuronal y la fórmula matemática que calcula el tiempo de operación de un relé de protección de sobre corriente con curva ANSI very inverse.

B. Caso 1 (i sobrecarga= 500A)

En el caso 1 se usa la corriente de sobrecarga con un valor de 500 A; a continuación, se presenta el error absoluto y relativo.

ERROR ABSOLUTO:

$$EA = |170.75 - 166.72| ms \quad (12)$$
$$EA = 4.03\ ms$$

ERROR RELATIVO:

$$ER = \frac{4.03}{166.72} \quad (13)$$
$$ER = 2.4\%$$

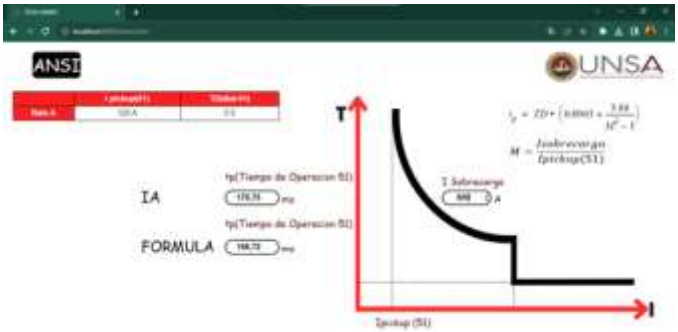

Fig. 19 Caso 1 (I SOBRECARGA= 500A)

C. Caso 2 (i sobrecarga= 800A)

En el caso 2 se usa la corriente de sobrecarga con un valor de 800 A; a continuación, se presenta el error absoluto y relativo.

ERROR ABSOLUTO:

$$EA = |93.79 - 92.8| ms \quad (14)$$
$$EA = 0.99\ ms$$

ERROR RELATIVO:

$$ER = \frac{0.99}{92.8} \quad (15)$$
$$ER = 1.06\%$$

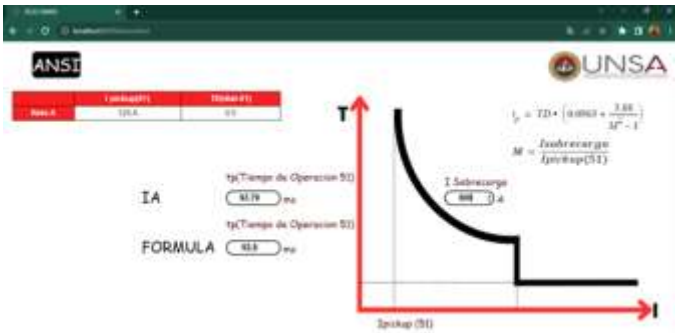

Fig. 20 Caso 2 (I SOBRECARGA= 800A)

D. Caso 3 (I Sobrecarga= 1000A)

En el caso 3 se usa la corriente de sobrecarga con un valor de 1000 A; a continuación, se presenta el error absoluto y relativo.

ERROR ABSOLUTO:

$$EA = |77.03 - 76.49| ms \quad (16)$$
$$EA = 0.54\ ms$$

ERROR RELATIVO:

$$ER = \frac{0.54}{76.49} \quad (17)$$

$$ER = 0.7\%$$

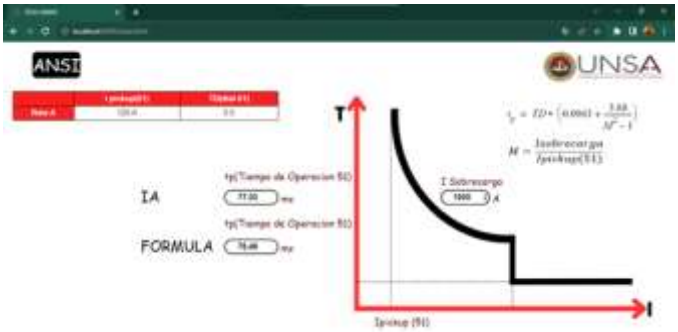

Fig. 21 Caso 3 (I SOBRECARGA= 1000A)

E. Análisis de resultados

De acuerdo a la tabla de errores absolutos y relativos del caso 1, 2 y 3. Se puede observar que los errores relativos varían.

Tabla N° 01
RESULTADOS DE LOS 3 CASOS

	Error Absoluto	Error Relativo
Caso 1	4.03 ms	2.4%
Caso 2	0.99 ms	1.06%
Caso 3	0.54 ms	0.7%

F. Discusión de resultados

Conforme al objetivo principal de esta investigación se a creado la interfaz web donde podemos realizar una comparación entre la predicción de una red neuronal y una fórmula matemática que calcula el tiempo de operación de un relé de protección de sobre corriente con curva ANSI very inverse. Se logró destacar que los errores relativos de los 3 casos son menores al 2.4%. Los errores relativos de la red neuronal aún pueden disminuir, esto se logra aumentando el número de épocas de 40000 a 300000, esto es posible con una computadora con aceleradores especializados como TPU o GPU, ya que la plataforma gratuita Google Colab que usamos en esta investigación nos permite realizar el entrenamiento hasta 40000 épocas con una duración de 15 horas de entrenamiento.

G. Aporte de los resultados

El resultado de esta investigación apoyará en el desarrollo de la inteligencia artificial en diferentes áreas profesionales y académicas.

También demuestra como transferir una red neuronal a una interfaz web que será de mucha utilidad en futuras investigaciones.

Se demostró cómo mejorar el error relativo ya que será de gran ayuda para mejorar el aprendizaje de una red neuronal.

IV. CONCLUSIONES

1. Se logró desarrollar una red neuronal con lenguaje de programación Python en la plataforma Google Colab, que realice la comparación entre la predicción de una red neuronal y una fórmula matemática que calcula el tiempo de

operación de un relé de protección de sobre corriente con curva ANSI very inverse.

2. Se pudo comprimir la red neuronal en un archivo ".json" para así poder trasladarlo en una interfaz web, utilizando JavaScript y TensorFlow.js.

3. Se creó una interfaz web que pueda utilizar la red neuronal comprimida en un archivo ".json"; utilizando lenguaje de programación HTML, CSS, JavaScript.

4. Al realizar los 3 casos de estudio se concluye que la inteligencia artificial realiza predicciones muy cercanas a las predicciones originales y para disminuir el error relativo se necesita de aceleradores especializados como TPU o GPU, ya que la plataforma gratuita Google Colab que usamos en esta investigación nos permite realizar el entrenamiento hasta 40000 épocas con una duración de 15 horas de entrenamiento.

REFERENCIAS

[1] Schweitzer Engineering Laboratories, Inc, *SEL-751 Relay - Feeder Protection Relay*. America: Schweitzer Engineering Laboratories, Inc, 2017, pp. 1-934. [Online]. Available: https://sertecrelays.net/wp-content/uploads/2019/02/751_IM_20170927.pdf

[2] IEEE Std C37.112-1996, *IEEE Standard Inverse-Time Characteristic Equations for Overcurrent Relays*, 1996.

[3] C. R. Mason, *Art & Science of Protective Relay*ing, 1st ed. Estados Unidos: General Electric, 2009.

[4] F. Berzal, Redes Neuronales & Deep Learning I. Universidad de Granada, 2019. [Online]. Available: https://deep-learning.ikor.org

[5] B. M. del Rio, *Redes Neuronales y Sistemas Borroso*s. Mexico D.F.: Alfaomega Grupo Editor, 2006.

[6] M. T. Hagan, *Neural Network Desig*n, 2014. [Online]. Available: http://hagan.okstate.edu/NNDesign.pdf

[7] Python Software Foundation, Python. [Online]. Available: https://www.python.org/. [Accessed: 2023].

[8] JavaScript.com. [Online]. Available: https://www.javascript.com/. [Accessed: 2023].

[9] HTML.com. [Online]. Available: https://html.com/. [Accessed: 2023].

[10] W3C, "CSS Overview," [Online]. Available: https://www.w3.org/Style/CSS/Overview.en.html. [Accessed: 2023].

[11] Google, "Google Colab," [Online]. Available: https://colab.research.google.com/. [Accessed: 2023].