INVESTIGACIÓN, TECNOLOGÍA E INNOVACIÓN Vol. 13 No. 13 JULIO - 2021 (45-54)
DOI: https://doi.org/10.53591/iti.v13i13.1093

Predicción del Composite Requerido en el Diseño de un Recipiente Toroidal Mediante una Red Neuronal Artificial

Prediction of the Composite Required in the Design of a Toroidal Vessel Using an Artificial Neural Network

Darwin Patiño Pérez
Miguel Botto-Tobar
Celia Munive Mora

Investigación
Tecnología e Innovación

INVESTIGACIÓN, TECNOLOGÍA E INNOVACIÓN Vol. 13 No. 13 JULIO - 2021 (45-54)

DOI: https://doi.org/10.53591/iti.v13i13.1093

Predicción del *Composite* Requerido en el Diseño de un Recipiente Toroidal Mediante una Red Neuronal Artificial

Prediction of the Composite Required in the Design of a Toroidal Vessel Using an Artificial Neural Network

Darwin Patiño Pérez[1], Miguel Botto-Tobar[2], y Celia Munive Mora[3]

Como citar: Patiño Pérez, D., Botto-Tobar, M., Munive Mora, C. (2021). *Investigación, Tecnología e Innovación*. 13(13), 44-54.
DOI: https://doi.org/10.53591/iti.v13i13.1093

RESUMEN

Contexto: Dentro del diseño de los recipientes toroidales, minimizar la cantidad de material, es muy importante para la reducción de costos de producción; los métodos convencionales que se usan para minimizar la cantidad de material, consumen mucho tiempo de computación cada vez que se ensaya sobre un nuevo recipiente. Las nuevas técnicas basadas en inteligencia artificial, garantizan la predicción de la cantidad mínima de material requerido para el diseño de un recipiente en el menor tiempo posible. **Método**: La metodología de predicción, está basada en un modelo de regresión lineal a través de una red neuronal artificial, que se implementa con el modelo de Keras de Python; en su primera fase, se maneja un dataset creado por un script con código APDL de ANSYS. **Resultados:** Un modelo de red neuronal artificial que aprendió a predecir la cantidad mínima de material, índices adecuados de exactitud y perdida del modelo. **Conclusiones:** Se obtuvo un mejor rendimiento del modelo, la partición del dataset en datos de *training y testing*, se obtuvo un nivel de precisión muy elevado que asegura la confiabilidad del modelo de *machine learning* frente a los tradicionales.

Palabras clave: Recipiente Toroidal, Inteligencia Artificial, Optimización, Machine Learning, Red Neuronal Artificial.

ABSTRACT

Context: Within the design of toroidal vessels, minimizing the amount of material is very important for reducing production costs; Conventional methods used to minimize the amount of material are computationally time consuming each time a new container is tested. New techniques based on artificial intelligence require the prediction of the minimum amount of material required for the design of a container in the shortest possible time. **Method:** The prediction methodology is based on a linear regression model through an artificial neural network, which is implemented with the Keras model of Python; In its first phase, a data set created by a script with ANSYS APDL code is handled. **Results:** An artificial neural network model that learned to predict the minimum amount of material, adequate accuracy and loss of the model. **Conclusions:** A better performance of the model was obtained, the partition of the dataset into training and testing data, a level of precision was obtained that ensures the reliability of the machine learning model compared to the traditional ones.

Keywords: Toroidal Vessel, Optimization, Machine Learning, Artificial Neural Network.

Fecha de recepción: Abril 28, 2021.

Fecha de aceptación: Julio 23, 2021.

[1] Ph.D, Universidad de Guayaquil, Ecuador. E-mail: darwin.patinop@ug.edu.ec
[2] Msc, Universidad de Guayaquil, Ecuador. E-mail: miguel.botto@ug.edu.ec
[3] BS, DeSales University, Pensilvania, EEUU. E-mail: cm3877@desales.edu

DOI: https://doi.org/10.53591/iti.v13i13.1093

INTRODUCCIÓN

En estudios de recipientes toroidales, se ha observado que existieron limitaciones que ocasionaron el uso de técnicas manuales para la elaboración de prototipos sometidos a presión (DeLay & Roberts, 2003), para el análisis y diseño de aquellos cuya estructura eran *shell* de *composite,* se usaron herramientas tecnológicas con las que se podían evaluar el rendimiento del material aplicando el método de elementos finitos y criterios de optimización como lo exponen(Vick & Gramoll, 2012) en su trabajo; en otros estudios se analizaron el comportamiento de diversos composites en las estructuras toroidales(Patiño & Corz, 2018) así como la predicción del inicio y evolución del daño(Barbero & Cosso, 2014) además de la elaboración del modelo de análisis progresivo de daños(Barbero & Shahbazi, 2017) que son un aporte a la hora de elaborar una técnica que permita determinar el daño en los recipientes toroidales. En los últimos estudios se aplicaron criterios de optimización condicionados por el criterio de fallo de tsai-wu (Patiño-Pérez & Corz-Rodríguez, 2019) experimentado en un recipiente toroidal de composite para el almacenamiento de GNC así como otro para el almacenamiento de Hidrogeno para vehículos(Patiño Pérez, Corz Rodriguez, & Mora, 2020) en los que el criterio de fallo en primera lamina condiciona la finalización del proceso de optimización que se utilice.

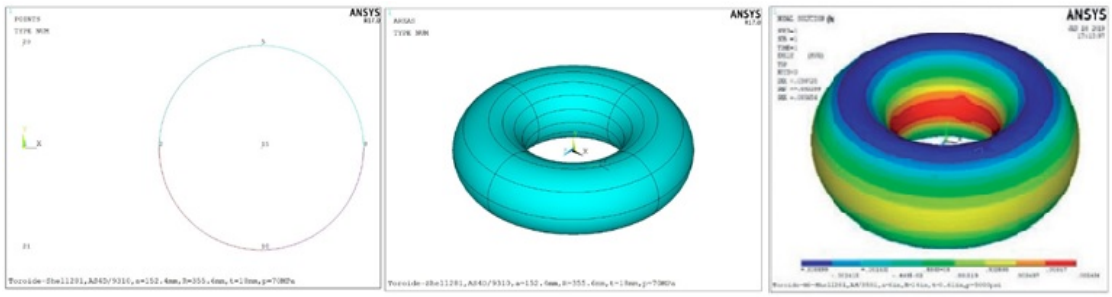

Figura 1. Toroide de Sección Recta Circular
Fuente: (Patiño & Corz, 2018)

Por la alta disponibilidad de recursos tecnológicos para inteligencia artificial(Pedrycz, Sillitti, & Succi, 2016), así como las presiones de predicción que ofrecen los algoritmos de aprendizaje automático o *machine learning*(Mathur, 2019), para el presente estudio se ha tomado una de las técnicas del subcampo del *deep learning*(Karlijn Willems, 2019) para la implementación de un modelo de red neuronal artificial (Yegnanarayana, 1994) que mediante aprendizaje supervisado(Brownlee, 2016), se ha entrenado para aprender a predecir el fallo en las áreas axiales de un toroide; se generó un dataset con las características más relevantes de varios modelos toroidales que fueron elaborados vía APDL.

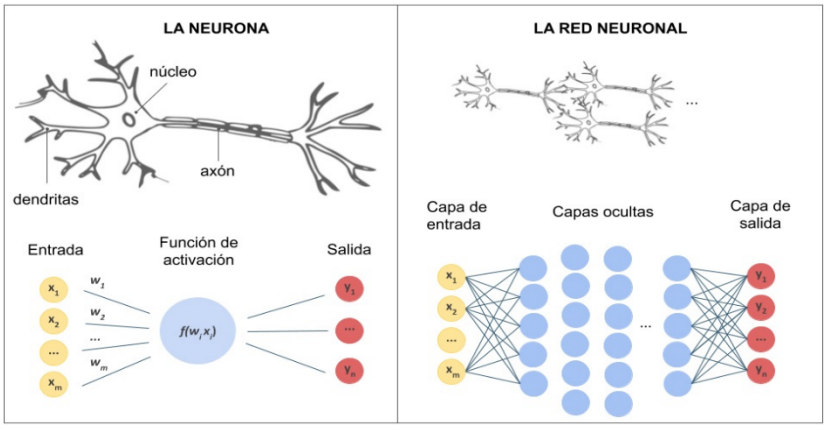

Figura 2. La Neurona y la Red Neuronal Artificial
Fuente: Propia

DOI: https://doi.org/10.53591/iti.v13i13.1093

MATERIALES Y MÉTODOS

Se ha empleado, un método de aprendizaje automático (*machine learning*) de dos fases, la primera fase comprende la creación del dataset, donde los toroide son obtenidos mediante el método de elementos finitos (MEF) con apartados de optimización sobre Ansys y codificado en APDL; se generaron dos especies de toroides, los que tienen fallo y los que no tienen fallo. La segunda fase, comprende la aplicación del modelo de Keras sobre Python, donde se crea el modelo de red neuronal artificial con más de una capa oculta la ubica dentro de la conceptualización de aprendizaje profundo (*deep learning*), este modelo está compuesto por la siguiente secuencia de trabajoo: procesamiento del dataset, creación del modelo de red neuronal artificial, compilación del modelo, entrenamiento del modelo, evaluación del modelo y realización de predicciones.

Creación del Dataset

En la predicción de la cantidad del material requerido, para el diseño del recipiente mediante redes neuronales artificiales(Agasi et al., 2018), se creó un dataset(Chapman et al., 2020) con datos provenientes de las simulaciones numéricas que se realizaron por medio del método de elementos finitos FEM, usando un script en APDL (ANSYS Parametric Design Language); se tomaron tres métodos disponibles en Ansys para simular las características principales de toroides optimizados con un índice de fallo aceptable, según el criterio de Tsai-Wu. El primer método que usado fue el FIRST, que es un método muy dependiente de la capacidad de cómputo del ordenador(Zou, Cao, Zhou, & Gu, 2020) en términos de GPU o CPU; el segundo método es fue el RAND, que obtiene simulaciones aleatorias y el tercer método es SUBP que está basado en el método de subproblema. El proceso general, para la construcción geométrica del toroide según ciertos requisitos necesarios, para determinar la deformación así como su índice de fallo, se lo codificó con el lenguaje paramétrico y se lo ejecuta en modo *batch* desde ANSYS(Lin et al., 2020), por medio de un script en APDL(toroF.txt).

La ejecución del APDL se la realiza vía línea de comando cmd al momento de ejecutarlo en modo *batch*.
! ansys170 -b -p1 355.6 -p2 152.4 -p3 30.0 -i toroF.txt -o salida.txt

Las características iniciales del toroide para la ejecución por FEM es: Toroide-Shell281,AS4D/9310,a=152.4mm,R=355.6mm,t=30mm,p=30MPa

El dataset(Moore, Saxon, Venkateswara, Berisha, & Panchanathan, 2019) creado tiene 11 campos (Tabla 1), (R) radio mayor del toroide, (a) radio menor del toroide, (P) presión a la que es sometido, (ANG1, ANG2, ANG3, ANG4) son los ángulos del apilamiento laminar de cada sección, (TK) el espesor del laminado, (MTSAI) el coeficiente de fallo, (FALLO) en esta columna se almacena (0 no tiene fallo , 1 si tiene fallo).

Tabla 1. Dataset/Dataframe

	R	a	P	ANG1	ANG2	ANG3	ANG4	TK	KG	MTSAI	FALLO
0	355.6	152.4	30	45.0000	45.00000	0.00000	90.0000	20.00000	65.039584	0.918235	0
1	355.6	152.4	30	45.0123	45.01190	6.19952	90.0000	19.98840	65.002040	0.924069	0
2	355.6	152.4	30	46.1771	46.16170	6.19929	90.0931	19.37240	62.998528	0.937293	0
3	355.6	152.4	30	45.9715	45.95710	5.98066	90.1354	19.19320	62.415912	0.941099	0
4	355.6	152.4	30	45.9706	45.95620	5.97960	90.1357	19.19340	62.416520	0.941111	0
...	...	...	...	...	...	...	...	...	...	...	...
2995	355.6	152.4	30	72.3327	4.50306	18.74930	141.7930	9.51986	30.958296	2.555310	1
2996	355.6	152.4	30	75.0031	16.08680	13.72560	120.9140	12.84840	41.782672	1.165660	1
2997	355.6	152.4	30	63.5878	5.89611	31.68290	112.0270	10.29390	33.475416	2.060520	1
2998	355.6	152.4	30	72.8063	11.32200	60.38380	106.6700	12.42530	40.406768	1.403290	1
2999	355.6	152.4	30	60.1618	10.86670	11.32800	132.0940	12.08130	39.288048	2.463010	1

3000 rows × 11 columns

Fuente: Elaboración Propia

DOI: https://doi.org/10.53591/iti.v13i13.1093

Procesamiento del Dataset

Previo a la carga de datos se definen una serie de bibliotecas de Python vía importación de las mismas, por lo que para el procesamiento de datos se usa Pandas, para el manejo de matrices y procesamiento numérico se usa Numpy y las de Keras para la creación del modelo de Red Neuronal Artificial.

```
import matplotlib.pyplot as plt
import numpy as np
from numpy import loadtxt
from keras.models import Sequential
from keras.layers import Dense, Activation
from keras import optimizers
import pandas as pd
```

El *dataset* contiene datos que plantean un problema de regresión lineal. Todas las variables de entrada que describen a cada toroide son numéricas. Esto hace que las mismas puedan procesarse de forma directa en el modelo de red neuronal artificial profunda; vía pandas se pasa del *dataset* al *dataframe* para el procesamiento interno de los datos.

```
df = pd.read_csv('datosTOR.csv', delimiter=';')
df.head().
```

El *dataframe* (df) contiene las características o variables de entrada al modelo, así como la variable de salida del mismo en sus respectivas columnas, puesto que las filas son datos de un toroide especifico, todas las variables de entrada al modelo se guardarán en una matriz X y la variable de salida se almacena en un vector y, por lo que del df se tomarán datos para entrenamiento(*training*) y datos para pruebas(*testing*).

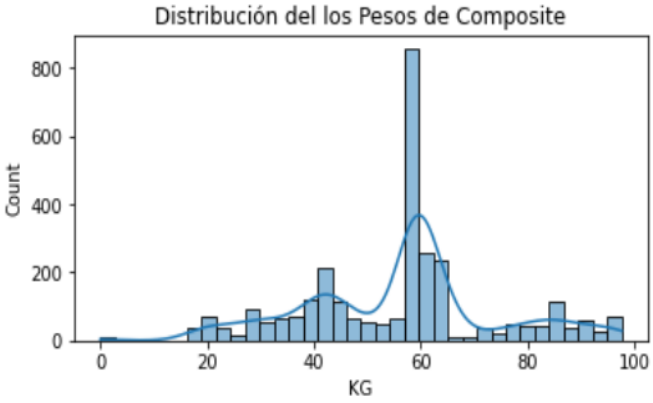

Figura 3. Modelo Secuencial de la RNA
Fuente: Elaboración Propia

Creación del Modelo de RNA

En la definición del modelo de red neuronal según (Figura-4), debemos tener en cuenta que del conjunto de datos se tomaran las 9 columnas, las primeras 8 columnas son las características de entrada mientras la 9na columna es el valor esperado o salida dentro del marco de datos a obtenerse.

DOI: https://doi.org/10.53591/iti.v13i13.1093

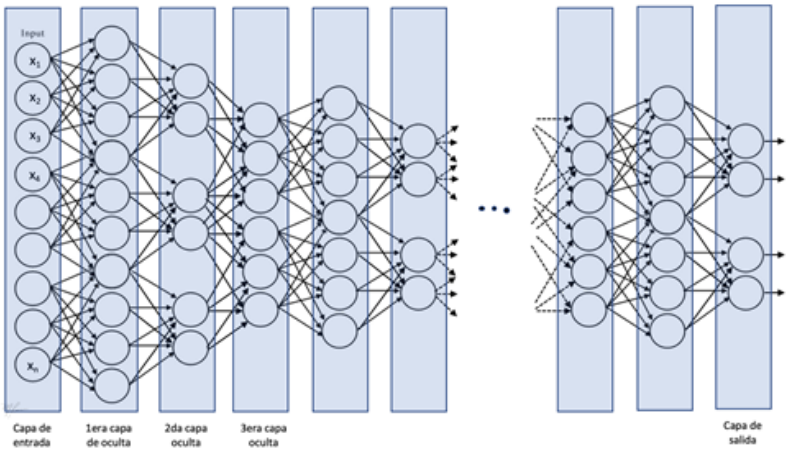

Figura 4. Modelo Secuencial de la RNA
Fuente: Elaboración Propia

Las redes neuronales se definen en Keras como una secuencia de capas. El contenedor de estas capas es la clase *Sequential*. Como primer paso se creó una instancia de la clase *Sequential* , luego se agregaron la capa de entrada, las ocultas y la de salida. La primera capa de la red debe definir el número de entradas que se esperan. Para un modelo de perceptrón multicapa, esto se especifica mediante el atributo "input_dim", el modelo de red tiene 8 entradas en la capa visible,256,128,16 neuronas en las 3 capas ocultas y 1 neurona en la capa de salida, cada capa con su correspondiente función de activación.

```
model = Sequential()
model.add(Dense(256, input_dim=8, kernel_initializer='normal',activation='relu'))
model.add(Dense(128, kernel_initializer='normal',activation='relu'))
model.add(Dense(16, kernel_initializer='normal',activation='relu'))
model.add(Dense(1, kernel_initializer='normal'))
```

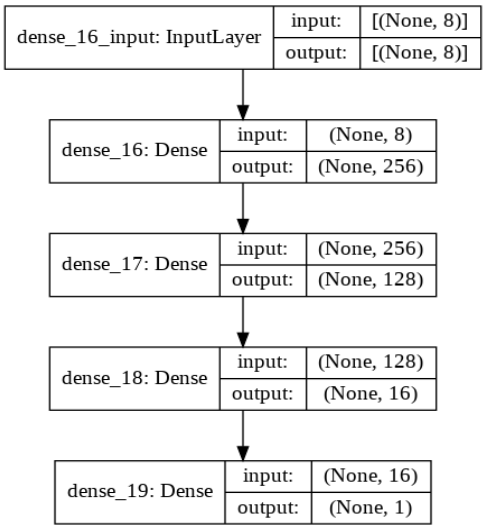

Figura 5. Modelo Secuencial de la RNA
Fuente: Elaboración Propia

Para la elección de la función de activación puesto que es lo más importante para la capa de salida, se consideraron los siguientes parámetros que son quienes definen el tipo de predicción.

DOI: https://doi.org/10.53591/iti.v13i13.1093

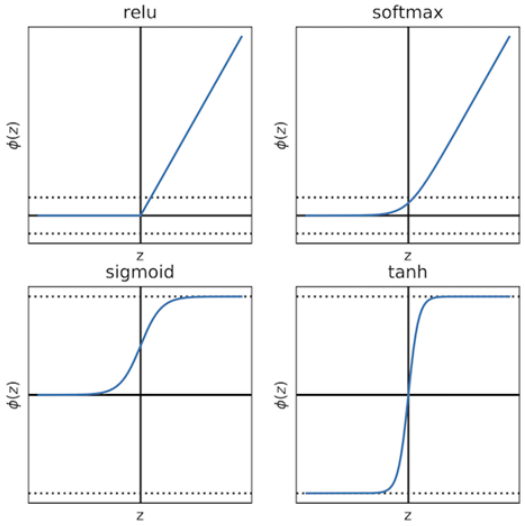

Figura 6. Funciones de Activación
Fuente: Elaboración Propia

• Regresión: función de activación lineal, o lineal, y el número de neuronas que coinciden con el número de salidas.
• Clasificación binaria (clase 2): función de activación logística, o sigmoide, una neurona en la capa de salida.
• Clasificación multiclase (> 2 clases): función de activación Softmax, o softmax, y una neurona de salida por valor de clase, asumiendo un patrón de salida codificado en caliente.

Compilación del Modelo de RNA

En la compilación se especificó el algoritmo de optimización que se utilizará para entrenar la red y la función de pérdida utilizada para evaluar la red que se minimiza mediante el algoritmo de optimización. Para la parametrización del método *compile* las opciones de configuración de las funciones de pérdida estándar para diferentes tipos de modelos predictivos que se tiene son:

• Regression: Error cuadrático medio.
• Binary Classification (2 class): Pérdida logarítmica, también llamada entropía cruzada o entropía cruzada binaria.
• Multiclass Classification (>2 class): Pérdida logarítmica multiclase o entropía cruzada categórica.

Los algoritmos de optimización de uso común que se pueden utilizar son:
• Stochastic Gradient Descent: requiere el ajuste de la velocidad y el impulso de aprendizaje.
• Adam: requiere el ajuste de la velocidad de aprendizaje.
• RMSprop: requiere el ajuste de la tasa de aprendizaje.

Finalmente, también se pueden especificar las métricas mientras se ajusta el modelo además de la función de pérdida. Generalmente, la métrica adicional más útil para recopilar es la *accuracy* para los problemas de clasificación, que para este caso en particular quedaría como:

model.compile(loss='mean_squared_error', optimizer='adam',metrics=['mean_absolute_percentage_error'])

Entrenamiento del Modelo de RNA

Después de compilar el modelo de red, se realiza el ajuste para adaptar los pesos en un conjunto de datos de entrenamiento, los mismo que se encuentran en la matriz de entrada "x_train" y en el vector de salida "y_train".

La red se entrena utilizando el algoritmo de *backpropagation* y se optimiza de acuerdo con el algoritmo correspondiente y la función de pérdida especificada al compilar el modelo. El algoritmo de *backpropagation* requiere que la red esté entrenada para un número específico de épocas. Cada época se puede dividir en grupos de pares de patrones de entrada-salida llamados lotes, los pesos se actualizan después de cada época y para

DOI: https://doi.org/10.53591/iti.v13i13.1093

este entrenamiento se ejecutaron 100 épocas y un tamaño de lote de 32. Estas configuraciones pueden elegirse experimentalmente mediante ensayo y error. El modelo se lo ha entrenado lo suficiente para que aprenda un buen (o suficientemente buen) mapeo de filas de datos de entrada a la clasificación de salida.

history=model.fit(x_train, y_train, batch_size=20, epochs=40,validation_data=(x_val,y_val))

Una vez que se ajusta, se devuelve un objeto histórico que proporciona un resumen del rendimiento del modelo durante el entrenamiento. Esto incluye tanto la pérdida como cualquier métrica adicional especificada al compilar el modelo, y que queda registrado en cada época.

Evaluación del modelo de RNA
Con la función evaluar () se obtienen dos valores, el primero será la pérdida y el segundo será la precisión del modelo, el conjunto de datos que se utilizan son x_test , y_test; aquí es importante mostrar la precisión el valor de la pérdida.
result = model.evaluate(x_test, y_test)

Realización de Predicciones
predictions = model.predict(x_test)

RESULTADOS

Se creó un modelo de Red Neuronal Artificial de tipo Secuencial basado en el modelo Keras de Python, esté modelo se ejecuta por encima de TensorFlow, que cuenta con la arquitectura adecuada para el manejo de redes neuronales, el mismo que garantiza un comportamiento apropiado cuando se utiliza un dataset con un gran volumen de datos.

De la creación del modelo de RNA:

Tabla 2. Capas de la RNA

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_16 (Dense)             (None, 256)               2304
_________________________________________________________________
dense_17 (Dense)             (None, 128)               32896
_________________________________________________________________
dense_18 (Dense)             (None, 16)                2064
_________________________________________________________________
dense_19 (Dense)             (None, 1)                 17
=================================================================
Total params: 37,281
Trainable params: 37,281
Non-trainable params: 0
_________________________________________________________________
```

Fuente: Elaboración Propia

Del Entrenamiento de la RNA: En la (Figura 8) se visualizan dos características importantes sobre el modelo:

1ra. A medida que aumenta el número de iteraciones, el error porcentual disminuye; esto indica que el modelo evoluciona hacia una dirección correcta.

2da. A partir de la iteración 5 inicia la disminución del error porcentual; esto significa que a medida que aumenta el número de iteraciones se produce la mejora en el desempeño del modelo.

DOI: https://doi.org/10.53591/iti.v13i13.1093

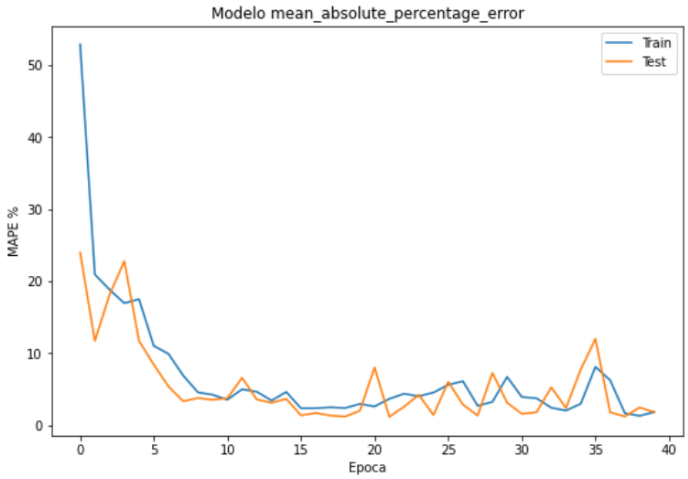

Figura 8. Tendencia MAPE
Fuente: Elaboración Propia

De la Evaluación de RNA: En cuanto a la evaluación de la función de pérdida del modelo.

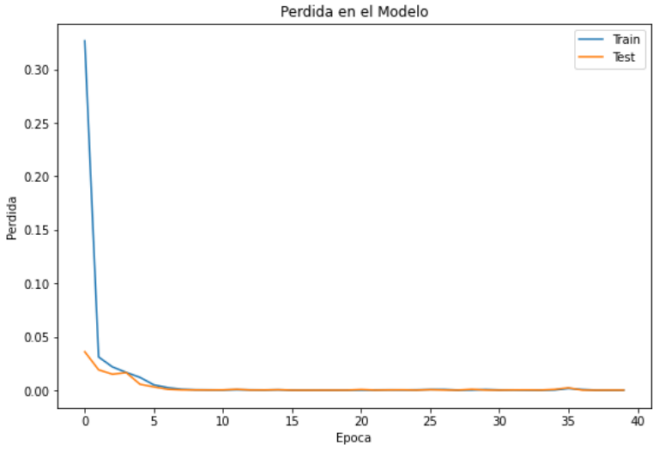

Figura 9. Tendencia Perdida del Modelo.
Fuente: Elaboración Propia

En la (Figura 9) de igual manera se pone de manifiesto el ajuste del modelo de acuerdo a los datos. A partir de la iteración época 5 las mejoras fueron relativamente grandes.

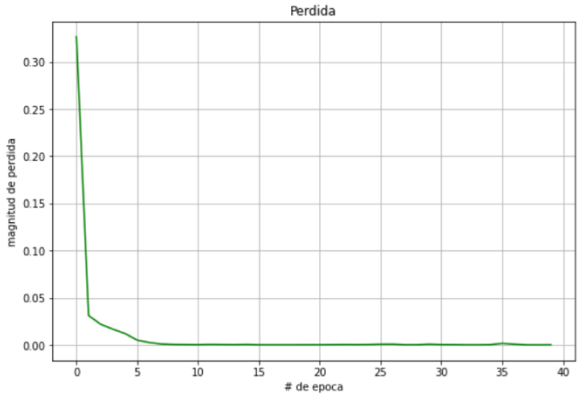

Figura 10. Tendencia Magnitud de Perdida
Fuente: Elaboración Propia

DOI: https://doi.org/10.53591/iti.v13i13.1093

De las Predicciones: Las predicciones realizadas con el 20% de los datos fueron registras tabla.

Tabla 3. Muestra de Predicciones Realizadas

	PesoReal	PredictedPeso
0	78.723688	78.558067
1	43.894560	43.641338
2	18.678824	18.292082
3	59.163568	59.065582
4	30.958296	30.725021
5	54.244088	53.916740

Fuente: Elaboración Propia

DISCUSION

A medida que aumentan las épocas de entrenamiento, la magnitud de la perdida se reduce, llegando a un punto en donde ya no ha mejora significativa (Figura 10).

No existe una gran diferencia entre el set de validación y el set de entrenamiento según (Figura 9), esto refleja que el modelo tiene un buen rendimiento tanto en entrenamiento como en pruebas.

La plantilla elaborada sobre Keras, se la puede utilizar para ajustar el modelo de regresión implementado en la ANN de aprendizaje profundo, sobre cualquier conjunto de datos.

Dependiendo del volumen de datos, se debería usar un determinado modelo de aprendizaje automático, para estimar la conveniencia de los tiempos de respuesta vs el modelo de red neuronal artificial.

CONCLUSIONES

Por la cantidad de capas ocultas, el modelo elaborado se lo define cono una ANN profunda; este tipo de redes neuronales artificiales funcionan muy bien cuando se tiene una buena cantidad de datos disponibles para aprender. Para pequeños conjuntos de datos de registros, es recomendable el uso de otros modelos ML supervisados.

La evaluación de la ANN profunda con el 80% de los datos para *training*, alcanzó una precisión del 97.847% y cuando se lo evaluó con el 20% de los datos de *testing*, alcanzó el 97.850%; por lo que el modelo tiene un reducido margen de diferenciación lo que lo vuelve confiable.

Las perdidas obtenidas tanto en el *training* como el en *testing* se fueron reduciendo a medida que aumentaban las épocas de entrenamiento, según se reflejan en la (Figura 9) lo que el modelo refleja un nivel muy elevado de confiabilidad.

Al haberse obtenido un modelo de red neuronal artificial, cuya efectividad en las predicciones es de aproximadamente el 98%, se puede concluir, que la nueva técnica para determinar el peso del material requerido para el diseño del toroide es muy efectiva y puede ser utilizada para reducir tiempos y costos de diseño.

DOI: https://doi.org/10.53591/iti.v13i13.1093

REFERENCIAS BIBLIOGRÁFICAS

1. Agasi, O., Anderson, J., Cole, A., Berthold, M., Cox, M., & Dimov, D. (2018). What is an Artificial Neural Network (ANN)? - Definition from Techopedia. *Techopedia*.
2. Barbero, E. J., & Cosso, F. A. (2014). Determination of material parameters for discrete damage mechanics analysis of carbon-epoxy laminates. *Composites Part B: Engineering*, *56*. https://doi.org/10.1016/j.compositesb.2013.08.084
3. Barbero, E. J., & Shahbazi, M. (2017). Determination of material properties for ANSYS progressive damage analysis of laminated composites. *Composite Structures*, *176*, 768–779. Retrieved from https://doi.org/10.1016/j.compstruct.2017.05.074
4. Brownlee, J. (2016). Supervised and Unsupervised Machine Learning Algorithms. *Understand Machine Learning Algorithms*.
5. Brownlee, J. (2017). How to Use the Keras Functional API for Deep Learning. *Machinelearningmastery*., (1).
6. Chapman, A., Simperl, E., Koesten, L., Konstantinidis, G., Ibáñez, L. D., Kacprzak, E., & Groth, P. (2020). Dataset search: a survey. *VLDB Journal*, *29*(1). https://doi.org/10.1007/s00778-019-00564-x
7. DeLay, T., & Roberts, K. (2003, November 1). Toroidal Tank Development for Upper-stages. Retrieved February 8, 2016, from http://ntrs.nasa.gov/search.jsp?R=20040084002
8. Karlijn Willems. (2019). (Tutorial) KERAS Tutorial: DEEP LEARNING in PYTHON - DataCamp.
9. Mathur, P. (2019). Machine Learning Applications Using Python. In *Machine Learning Applications Using Python*. https://doi.org/10.1007/978-1-4842-3787-8
10. Moore, M., Saxon, M., Venkateswara, H., Berisha, V., & Panchanathan, S. (2019). Say what? A dataset for exploring the error patterns that two ASR engines make. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, *2019-Septe*. https://doi.org/10.21437/Interspeech.2019-3096
11. Patiño-Pérez, D., & Corz-Rodríguez, A. (2019). Optimum design of a toroidal pressure vessel of composite material for gas (CNG) powered vehicles. *Dyna (Spain)*, *94*(5), 546–553. https://doi.org/10.6036/9096
12. Patiño, D., & Corz, A. (2018). Comparative Analysis of Toroidal Pressure Vessels of Composite by Finite Elements. *International Journal of Innovation and Applied Studies(IJIAS)*, *25*(1), 162–175. Retrieved from http://www.ijias.issr-journals.org/abstract.php?article=IJIAS-18-229-30
13. Patiño Pérez, D., Corz Rodriguez, A., & Mora, C. M. (2020). Diseño Optimo de un Recipiente a Presión Toroidal de Espesor Variable para Almacenar Hidrogeno en Automóviles Optimal Design of a Toroidal Pressure Vessel of Variable Thickness for Storing Hydrogen in Cars. *ECUADORIAN SCIENCE JOURNAL*, *4*(2), 94–100. https://doi.org/10.46480/esj.4.2.107
14. Pedrycz, W., Sillitti, A., & Succi, G. (2016). Computational intelligence: An introduction. In *Studies in Computational Intelligence*. https://doi.org/10.1007/978-3-319-25964-2_2
15. Vick, M. J., & Gramoll, K. (2012). Finite Element Study on the Optimization of an Orthotropic Composite Toroidal Shell. *Journal of Pressure Vessel Technology*, *134*(5), 1–7. https://doi.org/10.1115/1.4005873
16. Yegnanarayana, B. (1994). Artificial neural networks for pattern recognition. *Sadhana*, *19*(2). https://doi.org/10.1007/BF02811896
17. Zou, D., Cao, Y., Zhou, D., & Gu, Q. (2020). Gradient descent optimizes over-parameterized deep ReLU networks. *Machine Learning*, *109*(3). https://doi.org/10.1007/s10994-019-05839-6