Uso de redes convolucionales y recurrentes para predicción en sistemas de refrigeración domésticos

Diego Iván Perea Montealegre, Carlos Ivan Osorio Moreno, Gabriel Jeannot Viaña, Samir Hassan Ordóñez

Facultad de Ingeniería, Universidad Autónoma de Occidente

Cali, Colombia

—Resumen : En este proyecto, se lleva a cabo la creación y evaluación de modelos auto-regresivos utilizando arquitecturas de redes recurrentes y convolucionales, esto con el fin de dar vida a un sistema que permita eficientar el consumo eléctrico de los sistemas de enfriamiento para el hogar mediante el enfoque y la utilización de redes de inteligencia artificial, permitiendo además, dar una introducción y una solución de entrada mejorable en el largo plazo a una necesidad latente mediante métodos con la capacidad de escalarse en todo el campo de la eficiencia energética y los ciclos de uso en todos los aparatos eléctricos a nivel doméstico e industrial.

—Summary: In this project, the creation and evaluation of auto-regressive models is carried out using recurrent and convolutional network architectures, in order to give life to a system that makes it possible to make the electrical consumption of home cooling systems more efficient through the approach and use of artificial intelligence networks, also allowing for an introduction and an input solution that can be improved in the long term to a latent need through methods with the ability to scale throughout the field of energy efficiency and cycles. Use in all electrical appliances, domestically and industrially.

INTRODUCCIÓN

A medida que avanza la tecnología trae consigo soluciones a diversas situaciones que nos han permitido mejorar nuestra calidad de vida y/o del ecosistema. Pero con cada solución llegan otras situaciones más en forma de daño colateral. Una de ellas es la excesiva dependencia del servicio de energía eléctrica, y con ello el consumo masivo de este servicio que en su mayoría continúa proviniendo de fuentes de generación no sustentables.

El mayor problema de esta situación es que una parte de la energía consumida en el día a día es malgastada y desaprovechada debido al estilo de vida de la gente y la manera en cómo funcionan las cosas que nos rodean. Por ello desde esta materia, se ha buscado sacar partido de las nuevas tecnologías con el fin de aportar a la necesidad de generar dispositivos con mayor eficiencia energética.

Por tanto, haciendo uso de esta temática y enfocándonos en mejorar día a día el ecosistema que nos rodea, se presenta ‘AIr condIoTioner’, un nuevo modelo en el funcionamiento que presentan los aires acondicionados que existen actualmente. El cual pretende asegurar un mayor ahorro de energía en el proceso de enfriamiento de estos equipos a lo largo del día.

I. FUNCIONAMIENTO DEL SISTEMA

Este sistema, que permitirá eficientar el gasto de energía de los aparatos eléctricos destinados a propósitos de refrigeración (inicialmente dirigido a aires acondicionados, pero con un concepto escalable a otros sistemas de enfriamiento y otros ámbitos) funciona con la incorporación de un sistema de sensado de variables de temperatura y humedad del entorno en el que se encuentre (habitaciones, salas de estar, etc); a partir del procesamiento de estos datos por medio de un modelo de aprendizaje automático logrará predecir las condiciones climáticas que se avecinan, y a través de estos resultados enviados al aire acondicionado poder realizar un autoajuste para hallar la mejor curva de enfriamiento en un determinado tiempo y/o su manutención en una temperatura dada. Permitiendo al final, gracias al internet de las cosas, desplegar esta información al usuario a través de una aplicación web.

II. DISEÑO, ENTRENAMIENTO Y ANÁLISIS PRELIMINAR DE LOS MODELOS DEEP LEARNING

El inicio de esta fase empieza con la recolección de datos mediante sensores específicos que permitirán capturar las variables a utilizarse dentro del sistema, las cuales son temperatura y humedad, consideradas como las mediciones más importantes dentro del contexto de un sistema de enfriamiento, y mediante las cuales, a partir de su correlación, se lograra determinar un punto de equilibrio ideal para un óptimo funcionamiento del aire acondicionado.

Posterior a la captura de los datos, se da inicio a la fase de tratamiento de los datos, sección en donde se les hará pasar por diferentes "filtros" con el fin de obtener una información mucho más limpia, generalizada y fácil de procesar, entre los más importantes destaca el proceso de normalización, inherente a casi cualquier dataset, para poder tener en un rango definido el valor de los datos, eficientando el tiempo de procesamiento en el modelo de IA y el costo computacional requerido.

Una vez pre procesados los datos, estos se encuentran listos para ingresar al modelo de IA, el cual generará la extracción de características a partir de la generalización de estos y la identificación de patrones de comportamiento que permitan la predicción futura de dichas variables. Para ello, en este contexto, se hará uso de técnicas de procesamiento de datos secuenciales para predicción mediante el uso particular de dos arquitecturas de 'Deep learning', las cuales son redes convolucionales 1D y redes recurrentes RNN.

Seguidamente, se documentará el procedimiento, objetivos y resultados del entrenamiento de dos modelos de deep learning con el objetivo de realizar una predicción multivariable.

La finalidad académica de esta sección se enfoca en realizar una comparación entre dos diferentes modelos de deep learning y verificar mediante diferentes parámetros de evaluación, cuál de los dos se ajusta mejor al contexto específico.

Dichos modelos, como se mencionó anteriormente, difieren en la composición de la arquitectura característica de cada uno según su enfoque (RNN o CNN), sin embargo, el preprocesamiento de los datos es igual en ambos, por tanto, se generalizara esta parte del proceso.

A Proceso general [6] de preprocesamiento de los datos

1. Conexión e importación de librerías en Google Colab

```
import datetime, os
import tensorflow as tf
from tensorflow import keras
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from tensorflow.keras.layers import Conv1D, MaxPooling1D,
import matplotlib.pyplot as plt
```

Fig. 1. Importación de librerías y conexión a Drive.

2. Limpieza general de los datos

```
#Ruta del archivo CSV
nombre_archivo = '/content/gdrive/MyDrive/Universidad/Procesamiento

# Se lee el archivo CSV con la columna combinada
datos = pd.read_csv(nombre_archivo)
print(datos.columns)

# Se divide la columna combinada en varias columnas
datos[['id', 'id_nodo', 'temperatura', 'humedad', 'fecha']] = datos

# Se elimina la columna combinada
datos = datos.drop(columns=['id\tidnodo\ttemperatura\thumedad\tfecha

# Guarda el nuevo DataFrame en un nuevo archivo CSV
datos.to_csv('archivo_procesado.csv', index=False)

#como hay un solo valor nulo, se procede a eliminar manualmente la
indice_a_eliminar = 180

# Se elimina la fila con el índice especificado
datos = datos.drop(indice_a_eliminar)
```

Fig. 2. División de dataset por columnas y eliminación de valores NULL - string.

3. Generación de data frame con datos a usar

```
# Se toman las columnas a procesar del dataset (temperatura, humedad y fecha)
columnas_procesar = datos[['temperatura', 'humedad', 'fecha']]

# Se imprimen las primeras filas del DataFrame resultante
print(columnas_procesar.head())

# Se crean dataframes para fecha y otro para temperatura y humedad juntas, se co
TempHum = datos[['temperatura', 'humedad']]
TempHum['humedad'] = pd.to_numeric(TempHum['humedad'], errors='coerce')
TempHum['temperatura'] = pd.to_numeric(TempHum['temperatura'], errors='coerce')
print(TempHum)
```

Fig. 3. Generación de data frame temperatura-humedad.

4. Normalización de los datos

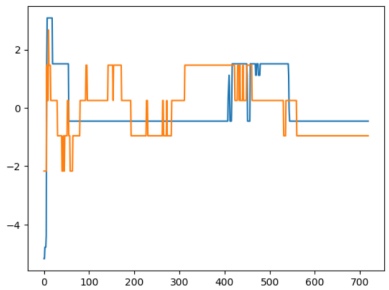


Fig. 4. Dataset normalizado.

5. *Se divide el dataset para entrenamiento y testeo*

```
#Se dividen los datos para el entrenamiento
InfTrain=TempHumN[0:500]
Dataset = keras.utils.timeseries_dataset_from_array(
data=InfTrain[:-10],
targets=InfTrain[10:],
sequence_length=10,
sequence_stride=1,
sampling_rate=1,
batch_size=1,
)
```

Fig. 5. Generación del dataset train.

6. *Se crea el enventanado y etiquetado de datos*

```
#Se crea el enventanado para los datos de entrenamiento con una
CantidadWindows=481
LongitudWindow=10

DatosVentanasInput=np.zeros((CantidadWindows,LongitudWindow,2))
DatosVentanasOutput=np.zeros((CantidadWindows,2))
```

Fig. 6. Enventanado de datos con la secuencia de ingreso al modelo.

En adelante, el proceso de entrada de los datos y entrenamiento se independizan según el tipo de modelo.

A. *Modelo 1: Red convolucional 1D*

1. *Se define la arquitectura del modelo CNN*

```
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_1 (Conv1D)           (None, 8, 32)             224

 max_pooling1d_1 (MaxPoolin  (None, 4, 32)             0
 g1D)

 flatten_1 (Flatten)         (None, 128)               0

 dense_2 (Dense)             (None, 10)                1290

 dense_3 (Dense)             (None, 2)                 22

=================================================================
Total params: 1536 (6.00 KB)
Trainable params: 1536 (6.00 KB)
Non-trainable params: 0 (0.00 Byte)
```

Fig. 7. Arquitectura del modelo CNN 1D.

2. *Diagramación y parametrización del modelo*

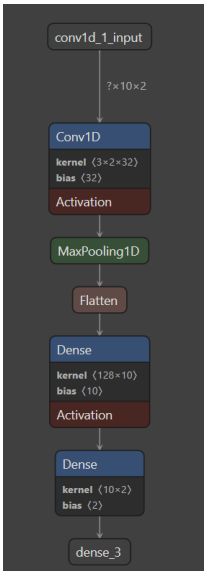


Fig. 8. Diagrama del modelo CNN.

3. *Resultado del entrenamiento*

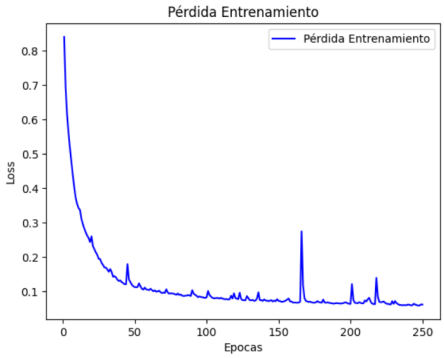


Fig. 9. Loss metric durante el entrenamiento.

4. *Cálculo del MSE con los datos de testeo*

```
MSE = modelo.evaluate(Xtest, Ytest)
print("Loss:", MSE[0])

7/7 [==============================] - 0s 2ms/step - loss: 0.0677 - mse: 0.0677
Loss: 0.06769178062677383
```

Fig. 10. Resultado del entrenamiento del modelo.

5. *Predicción del modelo y comparación de resultados:* Una vez obtenidos los resultados, se logra evidenciar que luego del entrenamiento, los datos predecidos se ajustan de forma adecuada a los datos reales, si bien es cierto que a lo largo de su recorrido presentan una gran cantidad de oscilaciones, estas aparecen debido a su mínima magnitud como una forma de ruido del modelo, que en líneas generales identifica de manera correcta el patrón. Se adjunta una gráfica comparativa

para más detalle (azul datos reales y rojo datos predichos).

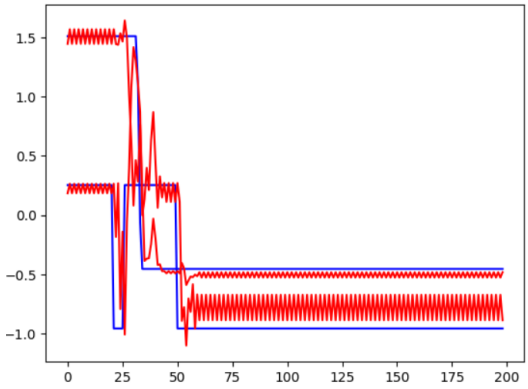


Fig. 11. Predicción de los datos.

En adelante, el proceso de entrada de los datos y entrenamiento se independizan según el tipo de modelo.

B. Modelo 2: Red recurrente RNN

1. Se define la arquitectura del modelo RNN

Layer (type)	Output Shape	Param #
simple_rnn (SimpleRNN)	(None, None, 5)	40
simple_rnn_1 (SimpleRNN)	(None, None, 10)	160
simple_rnn_2 (SimpleRNN)	(None, 5)	80
dense (Dense)	(None, 2)	12

Total params: 292 (1.14 KB)
Trainable params: 292 (1.14 KB)
Non-trainable params: 0 (0.00 Byte)

Fig. 12. Arquitectura del modelo RNN.

2. Diagramación y parametrización del modelo

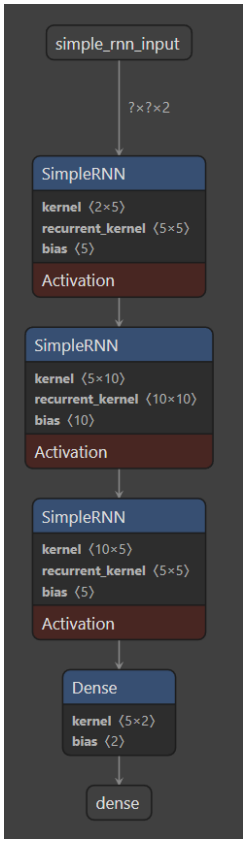


Fig. 13. Diagrama del modelo RNN.

3. Resultado del entrenamiento

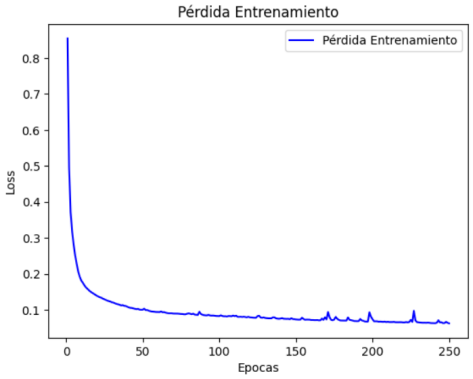


Fig. 14. Loss metric durante el entrenamiento.

4. Cálculo del MSE con los datos de testeo

```
MSE = modelo.evaluate(Xtest, Ytest)
print("Loss:", MSE[0])

7/7 [==============================] - 0s 4ms/step - loss: 0.0309 - mse: 0.0309
Loss: 0.030865568667650223
```

Fig. 15. Resultado del entrenamiento del modelo.

5. *Predicción del modelo y comparación de resultados:* A diferencia del resultado con el modelo CNN 1D, este modelo, debido a sus unidades con memoria recurrente, permiten obtener un comportamiento más fidedigna a la realidad, tanto numérica como visualmente se logra evidenciar un ajuste más preciso a la curva de los datos reales, resaltando principalmente la ausencia de tanta oscilación presentada en el anterior modelo. A continuación se adjunta una gráfica comparativa para más detalle (azul datos reales y rojo datos predichos).

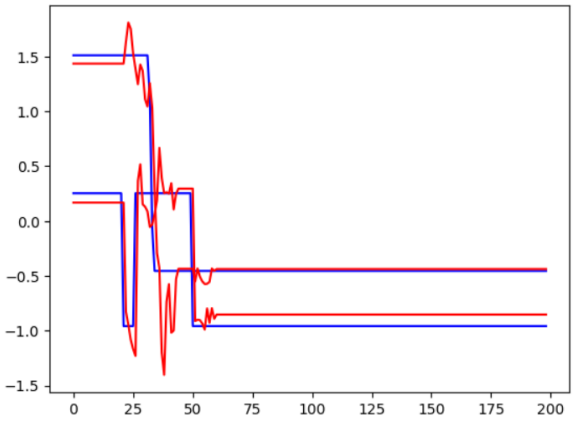


Fig. 16. Predicción de los datos.

III. ENTRENAMIENTO DE MODELOS EN EDGE IMPULSE Y DESPLIEGUE

Para iniciar con este proceso, se cuenta con los dos modelos entrenados previamente en el punto anterior; CNN1D y RNN. Partiendo del principio low-code que ofrece Edge Impulse, se crea un proyecto y se importa el dataset. Al ser un CSV, se hace uso de CSV Wizard para definir la estructura de este dataset antes de montarlo al proyecto. Utilizamos el siguiente dataset, siendo el mismo usado en el punto anterior, pero con la diferencia de que la fecha se ha convertido a un timestamp con el objetivo de evitar errores al momento de entrenar.

datasetIOT_transformed_single_timestamp.csv

ROW #	ID	IDNODO	TEMPERATURA	HUMEDAD	TIMESTAMP
# 1	1	1	24.1	59	1698621129
# 2	2	1	24.1	59	1698621134
# 3	3	1	24.2	59	1698621140
# 4	4	1	24.2	59	1698621145
# 5	5	1	24.2	59	1698621150
# 6	6	1	24.3	59	1698621156
# 7	7	2	25.8	62	1699725844

En este caso, se nota que la importación a CSV Wizard se hace de manera correcta. A continuación, realizamos distintos tipos de ajuste, empezando por la definición y organización del timestamp:

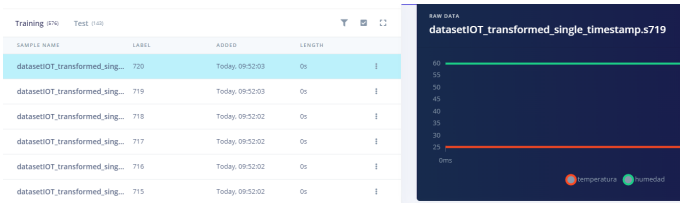
datasetIOT_transformed_single_timestamp.csv

TIME (MS.)	ID	IDNODO	TEMPERATURA	HUMEDAD
0	1	1	24.1	59
5	2	1	24.1	59
10	3	1	24.2	59
15	4	1	24.2	59
20	5	1	24.2	59
25	6	1	24.3	59
30	7	2	25.8	62

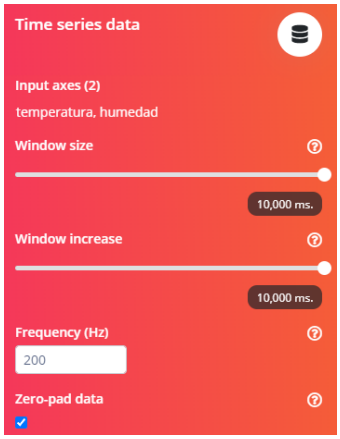
Una vez organizado lo relacionado con el timestamp, procedemos a definir el label, usando la variable “id” como label, obteniendo el siguiente resultado:

TIME (MS.)	LABEL (ID)	TEMPERATURA	HUMEDAD
0	1	24.1	59
5	2	24.1	59
10	3	24.2	59
15	4	24.2	59
20	5	24.2	59
25	6	24.3	59
30	7	25.8	62
35	8	26.2	61
40	9	26.2	61

Finalmente, importamos el dataset luego de hacer dichas configuraciones y obtenemos el siguiente resultado:

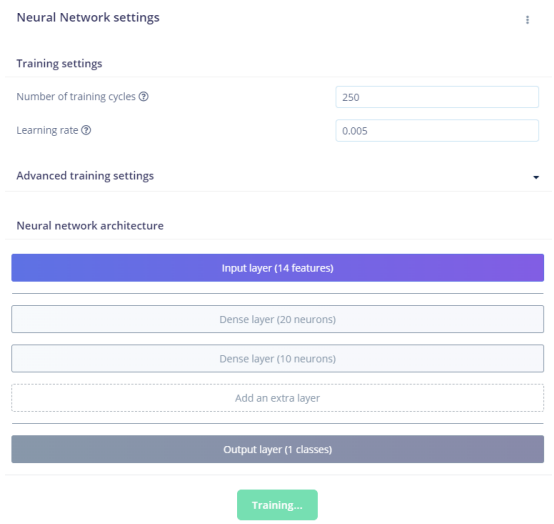


Se puede evidenciar una cantidad total de 719 samples para realizar el entrenamiento. Estos se dividen en 576 para Training, y 143 para Test. Una vez tenemos el dataset importado y configurado para una regresión, procedemos a realizar crear el impulse. Comenzamos definiendo un Time series data con 2 Input axes (temperatura, humedad).



Después, agregamos un bloque de procesamiento. Se agrega un bloque de Flatten, ideal para datos de movimiento lento como temperatura. Después, agregamos un bloque de aprendizaje, eligiendo Regression porque así se definió el primer modelo y por la lógica del proyecto. Se comienza a generar las features de Flatten y obtenemos el siguiente resultado:

Comienza el entrenamiento del modelo con las siguientes configuraciones:



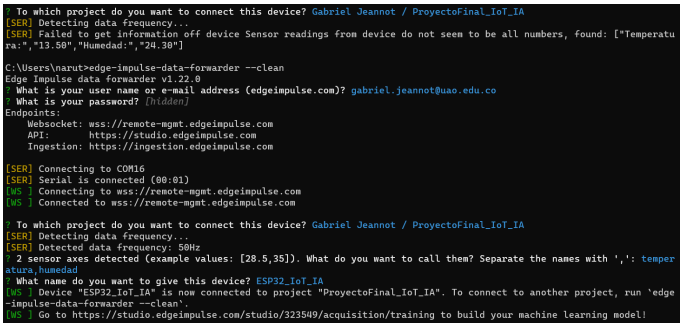
Finalmente, se entrena el modelo para su uso posterior una vez se realice la conexión desde Arduino para su prueba.

Ahora, se realiza la conexión con el ESP32, simulando el envío de la temperatura y humedad. El código utilizado para este proceso es el siguiente:

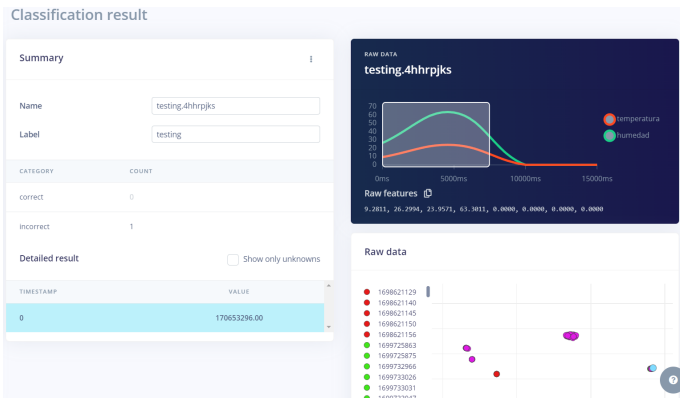
```
void setup() {
  Serial.begin(115200); // Inicia la comunicación serial a 115200 baudios
  while (!Serial) delay(10); // Espera a que la consola serial esté disponible

  Serial.println("Iniciando simulación de Sensor DHT11");
  delay(100);
}

void loop() {
  // Simula la lectura de temperatura y humedad
  float temperatura = random(100, 300) / 10.0; // Temperatura entre 10.0°C y 30.0°C
  float humedad = random(200, 900) / 10.0;     // Humedad entre 20.0% y 90.0%

  // Imprime solo los valores numéricos de temperatura y humedad separados por una coma
  Serial.print(temperatura);
  Serial.print(",");
  Serial.println(humedad);

  delay(20); // Mantiene la frecuencia de 50 Hz
}
```

Utilizando este código, podremos enviar datos a través de la conexión que se puede establecer con el comando: edge-impulse-data-forwarder --clean

Se utiliza el comando, se ingresa con el usuario y contraseña, posteriormente se selecciona el proyecto, en este caso se llama ProyectoFinal_IoT_IA, y se definen las dos variables a ser recibidas por Edge Impulse. Todo esto se puede evidenciar en la siguiente imagen:



Una vez tenemos la posibilidad de enviar datos, vamos directamente a Live Classification dentro de Edge Impulse para hacer una prueba real.

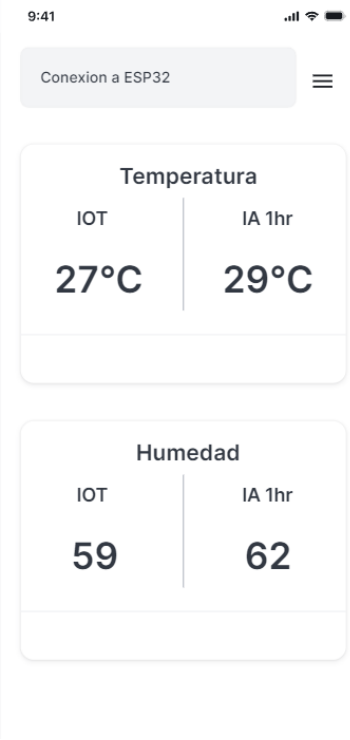


Evidenciamos que los resultados de la predicción no fueron positivos, y esto se debe a que el modelo arrojó resultados deficientes en su entrenamiento. Sin embargo, se puede concluir que, con un buen modelo, se puede establecer la conexión y realizar la predicción de forma óptima.

Se comparte a continuación el enlace a Edge Impulse en modo público: https://studio.edgeimpulse.com/public/323549/latest

IV. INTERACCIÓN DEL MODELO DESPLEGADO

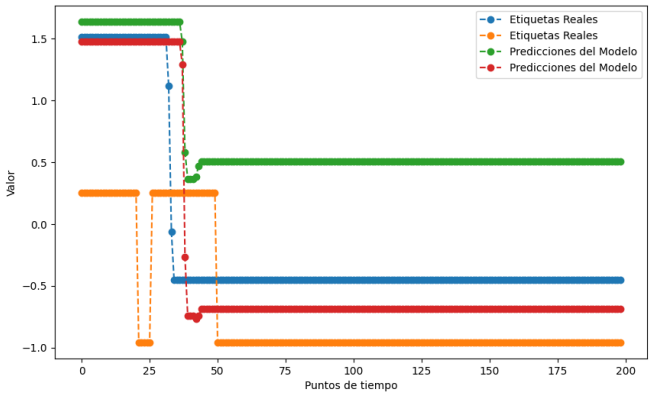
Con el modelo desplegado y conectado al ESP32 , los datos obtenidos por el sensor DHT11 de temperatura y humedad son enviados a la aplicación móvil y el modelo I.A. predecirá cuál será su temperatura y humedad en el tiempo entrenado por el modelo. En este ejemplo, sería en una una hora , esto sería de mucha utilidad para el usuario o para dicho objeto IOT con IA, de una forma clara y sencilla.



OPTIMIZACIÓN DEL MODELO

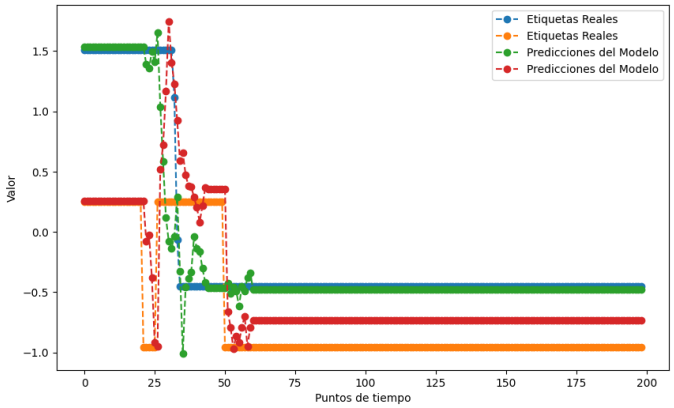
Para la optimización del modelo se usó tensor flow lite en donde se redujo el tamaño de los dos modelos en donde el modelo actual pesaba aproximadamente 24 kB , con la optimización del modelo en tensor flow lite se pudo reducir hasta un 5 kB aproximadamente.

A continuación se muestra la gráfica del modelo de RNN optimizado con TFlite comparándolo con los datos de testeo.



Se evidencia que se baja ligeramente la predicción del modelo pero dando buenos resultados en los datos de testeo X. Cabe aclarar que no se realizó la gráfica del testeo por temas de tiempo.

Con el modelo de CNN1D se muestra la gráfica del modelo optimizado TFlite comparándolo con los datos de testeo.



Se evidencia que se baja ligeramente la predicción del modelo pero dando buenos resultados tanto en los datos de testeo de X y Y, dando aclarar la buena forma de optimización del modelo para ser introducido en un dispositivo embebido como lo es el ESP32.

Con el modelo .tflite se procede a realizar el embebido en este caso hacia el ESP32 , para hacerlo hay que seguir estos pasos de este link de un video:https://www.youtube.com/watch?v=kZdIO82059E

IV. CONCLUSIONES

El desarrollo y la creación de diferentes librerías ya consolidadas en torno al mundo de la inteligencia artificial, específicamente del aprendizaje profundo, permite acercar este tipo de enfoques complicados a simple vista, a cada vez más personas con una facilidad técnica que se reduce a aprender únicamente la teoría base detrás de ellas, permitiendo dirigir

mayor atención a la recolección, diseño y análisis de datos sin limitaciones.

Si bien es sabido que el costo computacional de un método de aprendizaje profundo (DL) frente a uno de aprendizaje automático (ML) es más alto, la historia ha demostrado y el desarrollo tecnológico exponencial ha permitido reducir paulatinamente esta desventaja y consolidar cada vez más al campo del DL como uno de los enfoques con mayor potencial de crecimiento a futuro gracias a su funcionamiento que se potencia en la correlación del entorno; su integración con diferentes campos como IoT, visión computacional, Big Data; la confiabilidad que lo precede; así como el crecimiento y amplio margen de mejora en el largo plazo.

RECONOCIMIENTOS

Este documento tiene un enfoque práctico y académico que pretende ser de fácil entendimiento y cuenta con un nivel de profundidad relativamente superficial, pero no hubiese sido posible sin las enseñanzas y material impartido por el profesor Jesús A. López y el programa de Especialización en Inteligencia Artificial de la Universidad Autónoma de Occidente.

REFERENCIAS

[1] “La OMS destaca que el descuido de la salud bucodental afecta a casi la mitad de la población mundial”. World Health Organization (WHO). Accedido el 3 de noviembre de 2023. [En línea]. Disponible: https://www.who.int/es/news/item/18-11-2022-who-highlights-oral-health-neglect-affecting-nearly-half-of-the-world-s-population

[2] N. Pallarés Martínez, “Relación de la deficiente higiene bucal y la consecuente aparición de patologías bucales con las enfermedades cardiovasculares.”, Trabajo de grado, Univ. Jaume I, Castellón de la Plana, 2019.

[3] “Ni nos lavamos los dientes lo suficiente ni sabemos cuidar del cepillo”. El País. Accedido el 3 de noviembre de 2023. [En línea]. Disponible: https://elpais.com/elpais/2020/02/19/buenavida/1582130938_406935.html

[4] *Caras de los dientes | Dentistry student, Dental, Dentistry*. Accedido el 3 de noviembre de 2023. [Imagen]. Disponible: https://co.pinterest.com/pin/550987954360150 98/

[5] W. You, A. Hao, S. Li, Y. Wang y Bin Xia. “Deep learning-based dental plaque detection on primary teeth: a comparison with clinical assessments - BMC Oral Health”. BioMed Central. Accedido el 3 de noviembre de 2023. [En línea]. Disponible: https://bmcoralhealth.biomedcentral.com/articles/10.1186/s12903-020-01114-6

[6] Jesus A. Lopez "Procesamiento de datos secuenciales", códigos de clase 521376-EEA2, Facultad de ingeniería, Universidad Autónoma de Occidente, Octubre 2023.

[7] Edge Impulse Inc. *Edge Impulse*. (Data collect and processing).

[8] Google. *Google Colaboratory*. (Model training powered by Python).

[9] “Fine-tuning a model for music classification - Hugging Face Audio Course”. Hugging Face – The AI community building the future. Accedido el 2 de noviembre de 2023. [En línea]. Disponible: https://huggingface.co/learn/audio-course/chapter4/fine-tuning

[10]Introducing Whisper. (s.f.). OpenAI. https://openai.com/research/whisper