Universidad Zaragoza
1542

Trabajo Fin de Grado

Diseño e implementación de una Red Neuronal Artificial (RNA) para predecir la temperatura
Design and implementation of an Artificial Neural Network (ANN) to predict temperature

Autor

Javier Lasala Vidal

Directora

Dra. Dª Piedad Garrido Picazo

Escuela Universitaria Politécnica de Teruel

2021

A mi familia,
Sin ellos llegar hasta aquí no hubiera sido posible.

Tabla de Contenido

Índice de Figuras

Índice de Tablas

Resumen

El propósito de este Trabajo Final de Grado (TFG) consiste en crear una Red Neuronal Artificial (RNA) capaz de predecir la temperatura media que hará en el día de mañana.

Para conseguirlo, ha sido necesaria la obtención y preparación de un conjunto de datos meteorológicos adecuado para su entrenamiento, así como la posterior validación de la misma. La fuente de datos, AEMET OpenData, es un sistema para la difusión y reutilización de la información de la Agencia Estatal de METeorología, que a través de una API REST permite la difusión y reutilización de la información meteorológica y climatológica de la Agencia, cumpliendo con la Ley 18/2015, de 9 de julio, sobre reutilización de la información en el sector público.

El tipo de RNA implementado ha sido Long-Short Term Memory (LSTM), por la necesidad de trabajar con series temporales. En primer lugar, la creación de la RNA se ha llevado a cabo con la optimización por defecto, posteriormente, se ha añadido la optimización Adam y por último, se creó una nueva red neuronal con la optimización de los hiperparámetros a través de Keras Tuner. El motivo de crear diferentes redes neuronales con diferentes optimizaciones es para comprobar cuál de esas optimizaciones permite, en este caso, obtener la red neuronal mejor entrenada y por lo tanto, conseguir una predicción más acorde con la realidad.

Palabras clave

Red Neuronal Recurrente, LSTM, AEMET, predicción temperatura climática, MongoDB

Abstract

The purpose of the Final Degree Project t is to create an Artificial Neural Network (ANN) able to predict the average temperature it will do tomorrow.

To achieve this, it has been necessary to obtain and prepare an adequate set of meteorological data for training, as well as its subsequent validation. The data source, AEMET OpenData, is a system for the data dissemination and reuse of the State Agency of METeorology that through a REST API allows the dissemination and reuse of the Agency's meteorological and climatological information, complying with Law 18/2015, of July 9, on the reuse of information in the public sector.

The type of RNA implemented has been Long-Short Term Memory (LSTM) by the need to work with time series. First, the creation of RNA was carried out with the default optimization, subsequently, it added Adam optimization and finally, a new neural network is created with the optimization of hyperparameter through Keras Tuner.

The reason to create different neural networks with different optimizations is to check what these optimizations allows to obtain the best-trained neural network and, therefore to get a prediction more in line with reality.

Keywords

Concurrent Neural Network, AEMET, LSTM, MongoDB, climate temperature prediction

1. Introducción y Objetivos

El cambio climático es una realidad, por lo que la sociedad necesita una manera para poder actuar en consecuencia. La información y las predicciones climáticas sirven como base para la adopción de decisiones en infinidad de campos, la información precisa sobre la climatología permite salvar vidas, bienes y sustenta el crecimiento económico.

La idea de este TFG viene dada de la empresa Nologin Consulting S.L, en la que se realizaron las prácticas profesionales. En esta empresa se trabaja con boyas marinas dispersas desde Reino Unido hasta las Islas Canarias (ver figura 1). Estas boyas dan información acerca del tiempo meteorológico que se espera en la zona en la que están localizadas y sirven para que los barcos y aviones que transitan por la zona puedan anticipar sus acciones. Por un lado, este TFG persigue el objetivo de predecir la temperatura media de mañana con una técnica concreta del aprendizaje automático (en inglés Machine Learning (ML)), las RNA. Por otro lado, los Objetivos de Desarrollo Sostenible (ODS), protagonistas de este TFG, son [1]:

- Educación de calidad (Objetivo 4): garantiza una educación inclusiva, equitativa y de calidad y promover oportunidades de aprendizaje durante toda la vida para todos. En concreto, las metas 4.4 y 4.7
- Industria, innovación e infraestructura (Objetivo 9): industria, innovación e infraestructuras. En concreto, las metas 9.4, 9.5 y 9.b
- Acción por el clima. (Objetivo 14): adoptar medidas urgentes para combatir el cambio climático y sus efectos. En concreto, la meta 13.1

Pero, tal y como está desarrollado, la capacidad de predecir otras variables meteorológicas sería una tarea sencilla de implementar.

Figura 1: Boya de mar con instrumentación ambiental

Se ha creado y entrenado una RNA con datos meteorológicos desde el año 1968 hasta el día de hoy, del tipo Memoria a Largo-Corto Plazo (en inglés Long-Short Term Memory (LSTM)). En puntos posteriores de esta documentación se explicará más en concreto este tipo de RNA.

Los datos que se han utilizado para el entrenamiento de la RNA, han sido obtenidos de la Interfaz de Programación de Aplicaciones (API) de la Agencia Estatal de Meteorología (AEMET). La API

REST que se ha empleado es AEMET OpenData [2] que proporciona una gran variedad de tipos de datos al usuario, en este caso se ha utilizado “Climatologías diarias”, consiguiendo así, los datos almacenados de un lugar en concreto en un periodo concreto.

Para poder trabajar con toda la información obtenida y descargada a través de la API, se ha tenido que diseñar e implementar un almacén de datos (en inglés DataWarehouse (DW)), de tendencia NoSQL. El software que se ha elegido para soportar físicamente este diseño es MongoDB. Posteriormente, se ha llevado a cabo el preprocesado de los datos. Para el entrenamiento de la red neuronal se ha empleado el 80% de los datos obtenidos y 20% restante para la validación. Una vez se han normalizado los datos, se ha creado la RNA con un total de 3 capas: a) una capa de entrada, b) una capa oculta del tipo LSTM y c) una capa de salida. La cantidad de neuronas empleadas ha sido calculada por la librería Keras Tuner, que se encarga de optimizar los hiperparámetros.

A continuación, se ha procedido a verificar el diseño y entrenamiento de la RNA, ya que, aunque en el entrenamiento la pérdida sea muy pequeña, puede que no trabaje bien. Para la verificación del diseño se han empleado gráficas Q-Q (cuantil-cuantil) y el diagrama de Taylor, de los que se hablará y hará más hincapié en la sección 4 *(Evaluación de la red neuronal)*.

Para finalizar con la introducción, se van a comentar brevemente las secciones de las que consta esta memoria: a) el estado del arte, donde se hace referencia a investigaciones con un ámbito muy similar al de este TFG, b) análisis, diseño e implementación de la RNA, c) la evaluación de la RNA donde se realiza una valoración de los distintos modelos creados, d) las licencias software y documental, que se hubieran escogido para el trabajo realizado en el mundo real y, e) las conclusiones y trabajo futuro, donde se mencionan los aspectos más importantes del TFG realizado y el trabajo futuro que podría llevarse a cabo para su puesta en producción y posibles ampliaciones y mejoras sobre la predicción.

2. Estado del Arte

En esta sección se va a proceder a realizar una búsqueda sobre trabajos similares en los principales repositorios de artículos de investigación: Google Scholar, IEEEXplorer, Elsevier, Researchgate y ArXiv. De todas las búsquedas realizadas, se han seleccionado 6 artículos, que son las que se van a proceder a analizar en profundidad.

El primer artículo, *"Redes neuronales en la predicción de micro-clima en la Zona de estudio La Hechicera, Mérida- Venezuela"* [3], escrito por Misael Rosales, Cesar Augusto Mora Benavides y Carlos Guada, se habla de una RNA creada para realizar predicciones climatológicas en un corto plazo, con datos a intervalos de 15 minutos durante 454 días. Para llevarlo a cabo usaron patrones simulados que contenían variaciones diarias de la radiancia y la temperatura. Consiguieron que la RNA predijera el intervalo (0,1] usando solo como entrada un intervalo [-1,0] de la correspondiente derivada P´(x) del polinomio. Por último, en la fase de producción de datos reales, encontraron que la red era capaz de predecir la temperatura con un 5 % de error en el rango horario [12:15 a 06:15] pm, sólo con los datos de temperatura en el rango [6:00 a 12:00) am. Se considera viable el uso de (RNA) para la predicción de micro-clima a corto plazo, pudiendo extender su uso a otras localidades, lo que sería útil para el desarrollo de planes de prevención de desastres, períodos de siembra y en la predicción de oferta de energía en plantas eólicas y solares.

En otro estudio se encuentra, traducida al castellano, *"Aplicación de la Red Neuronal en la predicción de la temperatura: una revisión"* [4], realizado por Charles Johnstone y Emmanuel D. Sulungu. El objetivo fue revisar diferentes literaturas para evaluar la aplicabilidad de la RNA en la predicción de la temperatura. Estas RNA mejoran significativamente la predicción y precisión de las temperaturas. El rendimiento del modelo de la RNA varía según la naturaleza y la cantidad de datos de entrada utilizados en el entrenamiento de la red, el número de neuronas en la capa oculta, la arquitectura de una red, la función de transferencia y en el algoritmo de entrenamiento.

Siguiendo con el artículo *"Predicción de temperaturas usando los datos basados en una red neuronal de largo-corto plazo de memoria"* [5] escrito por Inyoung Park, Hyun Soo Kim, Jiwon Lee, Joon Ha Kim, Chul Han Song y Hong Kook Kim. Estos autores proponen un nuevo modelo de predicción de temperatura basado en el aprendizaje profundo, utilizando los datos meteorológicos reales que observaron. Para esto, necesitaron gran cantidad de datos de entrenamiento del modelo, pero que no fueran defectuosos. Sin embargo, encontraron una limitación en la recopilación de datos meteorológicos, ya que no era posible medir datos que se habían perdido. Además, las diferentes configuraciones de LSTM son investigadas con el propósito de que el modelo basado en LSTM pueda reflejar los rasgos de series de tiempo de los datos de temperatura. Por tanto, cuando se detecta que falta una parte de los datos, se restaura utilizando la función de refinamiento del modelo propuesto. Para finalizar, la propuesta basada en LSTM, el modelo de predicción de temperatura podía predecir la temperatura a través de tres tiempos: 6, 12 y 24 horas. Además, el modelo se amplió para predecir temperaturas futuras de entre siete a catorce días. El desempeño del modelo propuesto se mide por su raíz del error cuadrático medio (RMSE) y se compara con los RMSE de una red neuronal profunda feedforward o prealimentada, una red neuronal LSTM convencional sin cualquier función de refinamiento, y un modelo matemático utilizado actualmente por la oficina meteorológica en Corea. Además, la precisión de predicción del modelo propuesto es mayor que la del Sistema de Predicción y Asimilación de datos Locales (LDAPS) del Modelo Unificado (UM) para predicciones de temperatura en veinticuatro horas.

En *"Reconocimiento de patrones meteorológicos mediante técnicas estadísticas y neuronales"* [6] escrito por Miguel Irigoyen Mancho, se ha llevado a cabo una previsión de meteorología de diferentes parámetros (precipitación, temperatura…) para la ciudad de Pamplona a través del reconocimiento de patrones atmosféricos anteriores. Se definieron los patrones, se analizaron y se compararon los resultados estimados con los datos reales. Se utilizaron diferentes métodos estadísticos como la regresión lineal hasta la estimación utilizando redes neuronales para ver qué daba mejores resultados.

Para finalizar, se encuentra *"Redes neuronales para el posprocesamiento de pronósticos meteorológicos conjuntos"* [7], escrito por Stephan Rasp y Sebastian Lerch. Hablan sobre que las predicciones meteorológicas en conjunto requieren un posprocesamiento estadístico de errores sistemáticos para obtener unos datos fiables y con unas probabilidades precisas. Stephan y Sebastian proponen una alternativa flexible basada en redes neuronales que pueden incorporar relaciones no lineales entre predictores arbitrarios variables y parámetros de distribución de pronóstico que son aprendidos de forma automática basados en datos, en lugar de requiriendo funciones de enlace preespecificadas. Además, la red neuronal entrenada se puede utilizar para obtener la importancia de las variables meteorológicas, desafiando la noción de redes neuronales como cajas negras ininterpretables. El enfoque de los autores puede llevarse fácilmente a otros problemas de pronóstico y post procesamiento estadístico. Anticipan que los avances recientes en el aprendizaje profundo, combinado con las cantidades cada vez mayores de modelos y datos de observación, transformará el posprocesamiento de las previsiones meteorológicas numéricas en el futuro.

A continuación, se van a resumir algunas características técnicas, de los trabajos seleccionados, en la siguiente tabla:

	Tipo de red o de aprendizaje	Lenguaje o framework de programación	Lugar de procedencia de los datos
"Redes neuronales en la predicción de micro-clima en la Zona de estudio La Hechicera, Mérida-Venezuela"	Red perceptrónica multicapa entrenada con retropropagación	CNN Promoter Software	Estación climatológica Adcon Telemetry modelo addwave 733
"Predicción de temperaturas usando los datos basados en una red neuronal de largo-corto plazo de memoria"	LSTM	Keras & TensorFlow	Oficina Meteorológica de Corea del Sur (KMA)
"Reconocimiento de patrones meteorológicos mediante técnicas estadísticas y neuronales"	Técnicas estadísticas y neuronales	MATLAB	Meteo Navarra, Tu tiempo
"Redes neuronales para el posprocesamiento de	EMOS-gl, EMOS-loc, EMOS-loc-bst	EMOS framework, Python, Keras,	THORPEX In-

pronósticos meteorológicos conjuntos”	Red completamente conectada (FCN)	Tensorflow	teractive Grand Global Ensemble (TIGGE)

Tabla 1: Características técnicas de los trabajos seleccionados

En el caso de este TFG, al haber sido realizado al amparo de la empresa Nologin Consulting S.L, existían unos requisitos previos que se debían de cumplir como: el lenguaje de programación, el objetivo final o el tipo de red neuronal a implementar. Dentro del campo de ML, este proyecto está basado en Deep Learning (DL) o aprendizaje profundo, se ha implementado una RNA en Python del tipo LSTM, la cual se detalla más profundamente en puntos posteriores, con el objetivo final de predecir la temperatura media que hará en el día de mañana. Puesto que no se ha publicado nada similar o que cumpla con los requisitos establecidos para la realización de este TFG; el trabajo es original e inédito.

3. Análisis, Diseño e Implementación de la Red Neuronal Artificial (RNA)

Al tratarse de un proyecto de Inteligencia Artificial (IA) la fase de análisis no se va a centrar en un estudio de requisitos al uso, sino que se van a tener en cuenta, por un lado, los requisitos impuestos por Nologin Consulting, S.L.: a) lenguaje de programación Python, b) trabajar con una RNA recurrente (RNN) y c) como problema a resolver, la predicción de la temperatura de mañana. Por otro lado, investigar el tipo de aprendizaje y de red más adecuado, al problema a resolver, para lo que hay que tener claros los siguientes aspectos que se van a comentar con detalle en esta sección: a) elementos básicos, b) función de entrada, c) función de activación, d) función de salida, e) topología de RNA, f) diseño de RNA, g) Implementación de RNA y h) alojamiento del proyecto (GitLab Unizar).

3.1 Elementos básicos de RNA

Las redes neuronales artificiales se basan en el comportamiento de las neuronas biológicas. En estas neuronas biológicas que se encuentran en el cerebro se pueden distinguir tres partes: dendritas, axón y soma (ver figura 2). Las dendritas captan los impulsos nerviosos que emiten otras neuronas anteriores, los impulsos captados son procesados por el soma y son transmitidos a través del axón a neuronas contiguas [8].

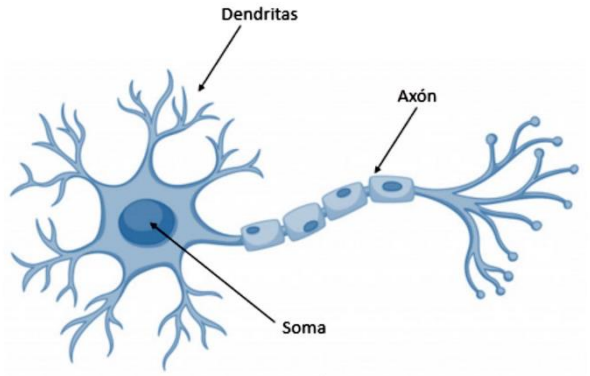

Figura 2: Estructura básica de una neurona

En el caso de las neuronas artificiales el "impulso nervioso" entre cada neurona está determinado por: el número de entradas de cada neurona multiplicado por los pesos asociados a cada entrada. Se procesa el valor obtenido mediante la función de activación y se envía como salida de la neurona. En las RNA los diferentes niveles de neuronas se llaman capas.

En la siguiente imagen se puede observar una RNA con cuatro capas, la primera capa o también llamada capa de entrada recibe los datos reales, la segunda y tercera capa son las capas ocultas debido a que se desconocen los valores de entrada y salida, cabe mencionar que la capa oculta puede estar constituida por cero o más capas, no tiene por qué estar compuesta por dos capas como es en este caso, la forma de interconectar estas capas puede ser de diferentes maneras lo que determina la topología de la RNA y, por último, la capa de salida que muestra el resultado [9].

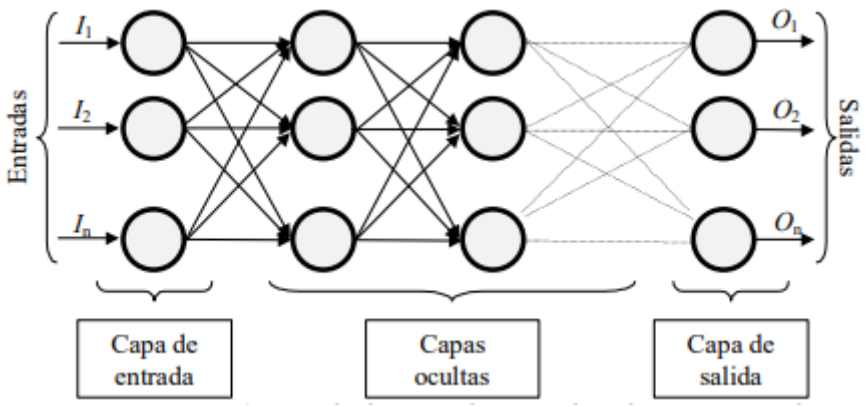

Figura 3: Estructura RNA

3.2 Función de entrada

Cada neurona que compone la RNA recibe muchos valores de entrada que son tratados como un solo valor. Para tratar los valores como uno solo, se logra a partir del vector de entrada.

$$input = X_{i-1}^{n} W_{ii} * X_i$$

3.3 Función de activación

Una neurona biológica puede tener dos estados, excitada (estar activa) o no excitada (inactiva). Las neuronas artificiales pueden tener diferentes estados, pueden tener dos estados como las neuronas biológicas, pero hay algunas que pueden tener cualquier valor dentro de un conjunto determinado.

La función de activación lo que hace es calcular el estado de actividad de una neurona, transformando el input, calculado en la función de entrada en un estado de activación, cuyo rango puede ir de (0, 1) o (-1, 1), siendo 0 o -1 el estado inactivo y 1 estado activo. Cada una de las capas, que forman parte de la RNA, tiene una función de activación [10].

3.3.1 Función de activación Relu

Es una función lineal que genera la entrada directamente si es positiva, de lo contrario genera cero. Es una de las funciones más utilizadas para las capas ocultas, es fácil de implementar y evita alguna limitación de otras funciones de activación como la Sigmoidea y Tangente Hiperbólica. Es menos susceptible a los "gradientes desaparecidos", es decir, en algunos casos, el gradiente se acerca a valores tan pequeños que impide que el peso cambie su valor, lo que puede impedir que la RNA continúe con el entrenamiento y, por lo tanto, aprendiendo.

$$f(x) = \max(0, x) = \begin{cases} 0 \; for \; x < 0 \\ x \; for \; x \geq 0 \end{cases}$$

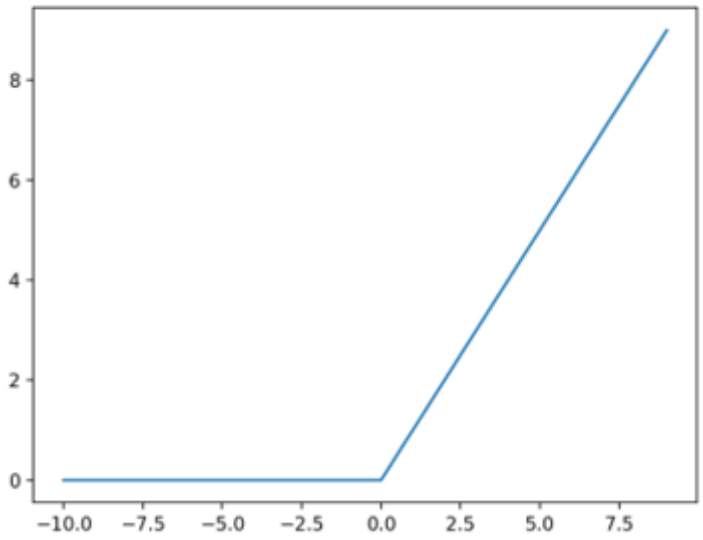

Figura 4: Función de activación Relu

3.3.2 Función de activación Sigmoidea

Toma cualquier valor real como valor de entrada y de salida en el rango de (0, 1), cuanto mayor es la entrada, más cerca estará el valor de salida a 1, mientras que ocurre lo contrario con valores de entrada menores.

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

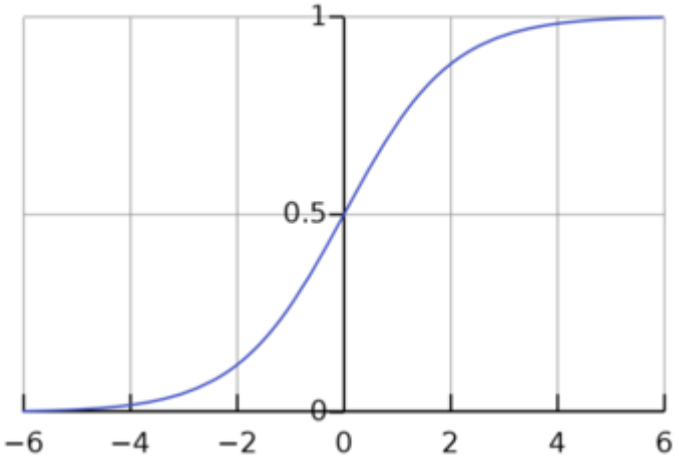

Figura 5: Función de activación Sigmoidea

3.3.3 Función de activación Tangente Hiperbólica

La función de activación también llamada tangente hiperbólica o Tanh, es una función muy similar a la ya comentada función Sigmoidea.

Toma cualquier valor real como valor de entrada y de salida en el rango (-1, 1), cuanto mayor es la entrada, más cerca estará el valor de salida a 1, ocurriendo lo contrario con valores de entrada más pequeños.

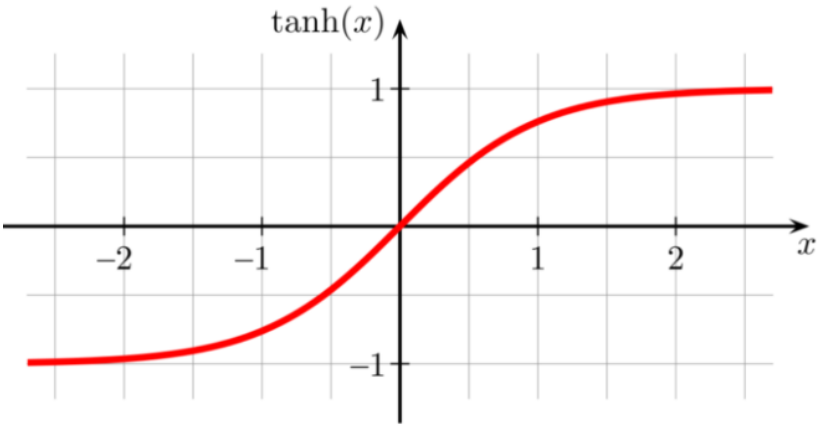

Figura 6: Función de activación Tangente Hiperbólica

3.4 Función de salida

Determina qué valor es transferido a la siguiente neurona o se da como resultado final. Como cualquier valor no está permitido para la entrada de una neurona, la salida viene en un rango de (0, 1) o (-1, 1). En el caso de este TFG no se ha utilizado ninguna función de salida, el valor obtenido como salida es pasado a la siguiente neurona.

3.5 Topología de RNA

Se van a explicar una serie de RNA diferentes que pueden ser: monocapa, multicapa, recurrentes o convolucionales (en inglés Convolutional Neural Network (CNN)). En este TFG se ha implementado una RNA recurrente, a petición de la empresa Nologin Consulting S.L. por lo que se va a hacer más hincapié en este tipo.

3.5.1 Red neuronal monocapa

Se corresponde con el tipo de RNA más simple, formada por una capa de neuronas. La información que entra a través de la capa de entrada va directamente a la capa de salida [11].

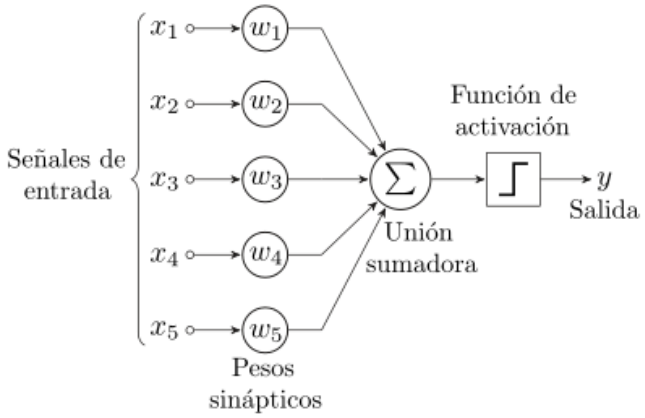

Figura 7: Diagrama de un perceptrón con 5 entradas

3.5.2 Red neuronal multicapa

La diferencia con el tipo de red neuronal monocapa se encuentra en que el tipo anterior solo está formado por la capa de entrada y la capa de salida. Este tipo de red contiene un número de capas intermedias, llamadas capas ocultas, entre la capa de entrada y la capa de salida, tal y como se puede ver en la siguiente imagen [11].

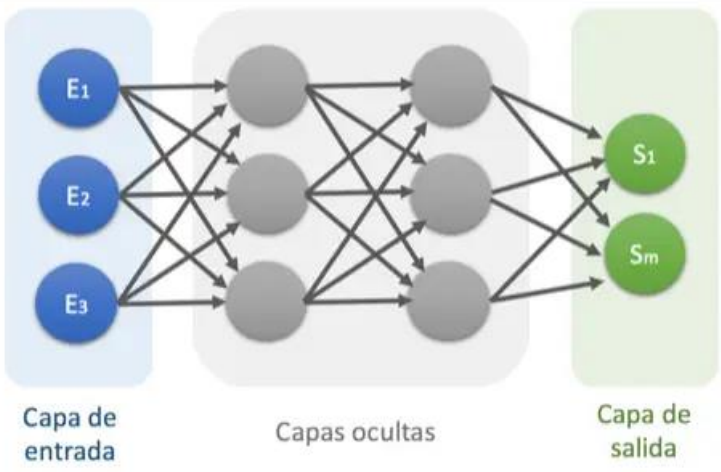

Figura 8: Estructura RNA multicapa

3.5.3 Red neuronal recurrente

Las RNN como su propio nombre indica, son parecidas a las redes neuronales multicapa, pero poseen conexiones "hacia atrás", lo cual permite una retroalimentación entre las neuronas dentro de las capas.

Si se imagina una RNN compuesta por una sola neurona en cada instante del tiempo, la cual recibe la entrada de la capa anterior y además su propia salida, se obtiene una representación gráfica como la siguiente.

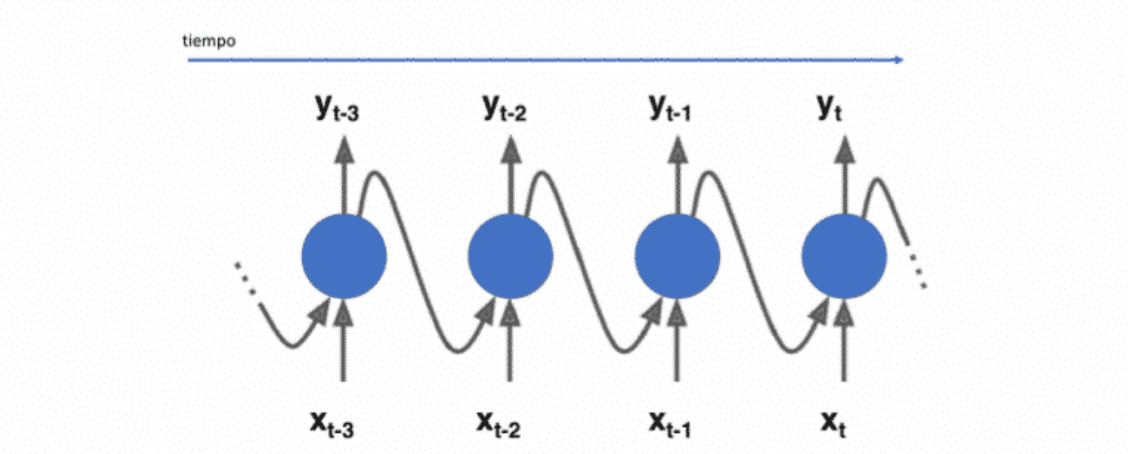

Figura 9: Una única neurona RNN en el tiempo

La parte de una neurona que preserva el estado a través del tiempo se llama memory cell. Principalmente es esta parte la que hace que este tipo de redes sean muy adecuadas para problemas de aprendizaje automático en el que se utilizan datos secuenciales, como en la traducción automática, modelado del lenguaje o reconocimiento de voz, ya que es mucho más importante la secuencia de datos que el contenido espacial.

3.5.4 Long-Short Term Memory

Habiendo introducido las RNN ahora se va a presentar un tipo de ellas, las LSTM, las cuales pueden recordar sus entradas durante un periodo de tiempo largo conteniendo la información en la memoria [12][13][14][15].

A través de los pesos, que son aprendidos con el paso del tiempo, se decide qué información es útil almacenar en la memoria o qué información hay que desechar, este proceso se realiza abriendo o cerrando las puertas que contienen cada neurona LSTM. Tal y como se acaba de mencionar, cada neurona contiene tres puertas de información con las que se va a decidir qué información se va a eliminar o mantener, estas puertas son: puerta de entrada (input gate), puerta de olvido (forget gate) y puerta de salida (output gate). Gracias a estas puertas se consiguen reducir los problemas de "*Vanishing Gradients"* o "gradientes desaparecidos" ya que los mantiene lo suficientemente empinados como para que la red pueda seguir con el entrenamiento y continuar con el aprendizaje.

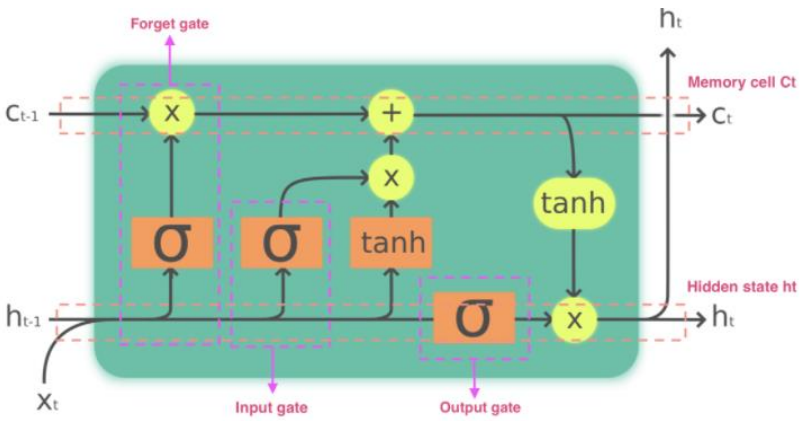

Figura 10: Neurona LSTM [16]

3.6 Diseño de RNA

Teniendo claro cómo funciona una RNA y los tipos que existen se va a pasar a comentar el diseño que se ha implementado para llevar a cabo este TFG.

En puntos anteriores se ha hablado de que este TFG se ha desarrollado con Nologin Consulting S.L., consultoría internacional con origen Aragonés. Ya que la idea es que esta RNA pueda ser utilizada para otros posibles proyectos de la empresa, en los que se necesite algo por el estilo, se ha utilizado Python como lenguaje de programación, Keras [17] y Tensorflow [18] para su implementación. Requisitos marcados por la empresa en su punto de partida.

El motivo de la creación de esta red neuronal es predecir la temperatura media que hará en el día de mañana, para lo que dentro del campo de ML, se ha empleado las redes neuronales por petición de la empresa, dentro de las redes neuronales, se ha empleado el tipo LSTM porque se va a trabajar con series temporales y nos interesa que la RNA que se ha creado tenga memoria, lo cual se consigue como se ha comentado en el punto anterior, con una RNN.

Para el entrenamiento de la RNA se necesita información, para ello hay que utilizar un sistema de difusión y reutilización de la información meteorológica. En este caso, se ha utilizado AEMET OpenData, para obtener la información necesaria. Como se trata de predecir el tiempo meteorológico, cuantos más datos se tengan, mejor se podrá entrenar la red neuronal.

3.6.1 Arquitectura

En la figura siguiente, se puede ver que en la parte de AEMET OpenData se necesita una API Key para poder obtener los datos. Se necesita registrar un correo electrónico en el que se recibe esa API Key, hay que tener en cuenta que la clave que se recibe dura un tiempo limitado, aproximadamente 3 meses. Mediante un script desarrollado en Python se obtienen los datos necesarios para entrenar la red neuronal.

Figura 11: Estructura del programa desarrollado

3.6.2 AEMET OpenData

AEMET OpenData es una API REST desarrollada por AEMET que permite la difusión y la reutilización de la información meteorológica y climatológica de la agencia [19].

Para poder acceder a AEMET OpenData, es necesario solicitar una API Key [20]. Un API Key es un identificador, mediante el cual se contabilizan e imputan los accesos que un usuario realiza a la API.

Existen dos formas de acceso para la obtención de datos que son las siguientes:

- Acceso general: El acceso para el público en general tiene como finalidad el permitir el acceso a los datos para usuarios de una manera amigable.
- Acceso desarrolladores: esta interacción se caracteriza por la posibilidad de ser periódica e incluso programada, realizada a través de un API destinada a un sistema informático, no se realiza a través de interfaces amigables y permite a desarrolladores incluir los datos de AEMET en sus propios sistemas de información.

La API de AEMET facilita una gran cantidad de datos que se pueden obtener, desde redes especiales como datos de radiación global, directa o difusa o perfiles verticales de ozono hasta predicciones normalizadas en texto o predicciones específicas.

Lo primero que se ha hecho para poder implementar OpenData en el TFG, ha sido aprender cómo funciona y qué datos se necesitan obtener. En el caso de este proyecto, los datos obtenidos de la API han sido "Valores climatológicos" → "Climatologías diarias". Utilizando el acceso general se puede ver qué datos son necesarios para rellenar el formulario, porque se necesita seleccionar una provincia, seleccionar una estación meteorológica, una fecha de inicio y una fecha final. Si por algún motivo se quiere una gran cantidad de datos, hay que tener en cuenta que la API sólo permite obtener datos en un rango de 5 años.

Figura 12: Captura de pantalla del acceso general de AEMET

Cuando se quiere obtener la información, va a aparecer una ventana emergente. En esa ventana aparecen una serie datos, pero las 2 URLs que aparecen son las que interesan:

- URL de datos: URL que contiene todos los días con sus datos correspondientes en cada campo comprendidos entre la fecha de inicio y la fecha final que se requieren para la red neuronal en este caso. Proporciona los datos en un fichero con formato JSON.
- URL de metadatos: URL que contiene todos los campos que tiene el fichero JSON explicando a qué hace referencia ese campo y qué tipo de datos contiene (string, float, int...)

3.7 Implementación de RNA

Sabiendo que una RNA es un grupo interconectado de nodos, similar a la red de neuronas en un cerebro biológico, donde: a) cada nodo representa una neurona artificial, b) cada flecha representa una conexión desde la salida de una neurona a la entrada de otra y c) cada nodo está conectado con otros a través de unos enlaces donde el valor de salida de la neurona anterior es multiplicado por un valor de peso, para su creación hay que seguir una serie de pasos: análisis de datos, preprocesado de los mismos, creación de la arquitectura de la RNA, compilación de la RNA, ajuste del modelo y evaluación de la RNA.

Una vez se tiene claro el funcionamiento de OpenData, se comienza con el análisis de datos. Lo que se ha hecho para llevar a cabo el proceso de recolecta de datos ha sido utilizar el acceso para desarrolladores que se puede encontrar en “Documentación AEMET OpenData. HATEOAS”. En esta parte se ofrece al desarrollador la consulta ya construida del tipo de dato que se quiera obtener, de esta forma, lo único que debe hacer a la hora de desarrollar la aplicación es acabar de rellenar la consulta con los datos deseados. En la figura siguiente, se puede observar un ejemplo de consulta que se ofrece y en el cual sólo se debe cambiar los datos "fechaIniStr", "fechaFinStr" e "idema".

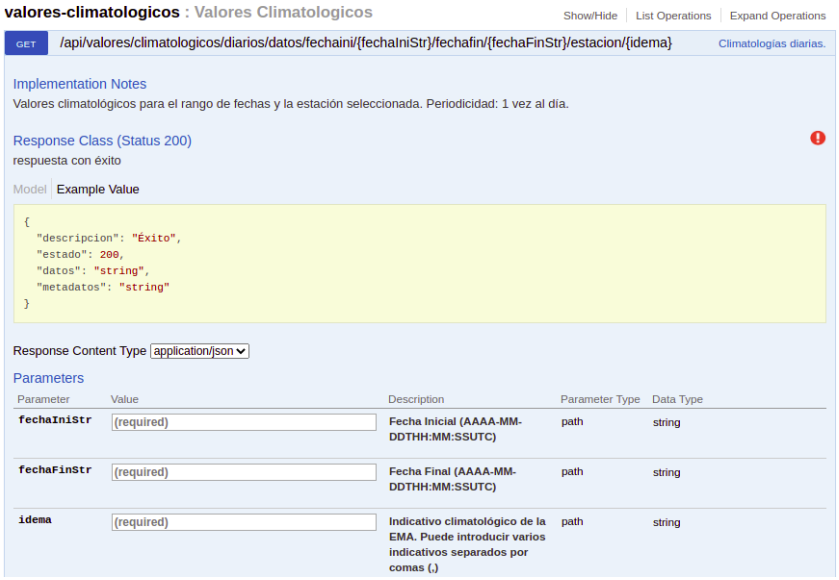

Figura 13: Captura de pantalla del acceso para desarrolladores de AEMET

En el siguiente ejemplo de consulta construida en modo desarrollador en la API se ha puesto como ejemplo de fecha inicial 01-01-1968, fecha final 31-12-1971 e idema 9434. El parámetro *idema*, tal y como se puede observar en la figura superior, es el identificador de la estación de la cual se están obteniendo los datos.

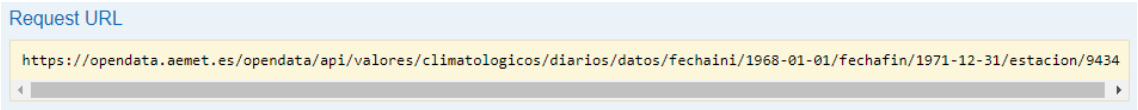

Figura 14: Consulta URL construida en modo desarrollador

La obtención de datos se ha realizado en Python, por lo que esta dirección URL se ha implementado en un script que descarga los datos de cada estación, comprendidos entre la fecha inicial y final, en este caso la fecha inicial de todos los datos es 01-01-1968 y la fecha final 5 años después. Tras obtener este bloque de datos se actualizan las dos fechas para obtener el siguiente bloque de datos, la siguiente fecha inicial es el día siguiente a la fecha final del bloque anterior y así sucesivamente hasta llegar a la fecha actual.

NoSQL es un almacén de datos, no usa SQL (en inglés Structured Query Language) como lenguaje principal de consulta y los datos almacenados no requieren de estructuras como tablas.

Con frecuencia, los repositorios de información NoSQL se clasifican según la forma que tienen de clasificar los datos, formando categorías como clave-valor, BD documentales o BD orientadas a grafos. Fueron creciendo cuando empresas como Google, Amazon o Facebook no podían realizar el tratamiento de datos con Sistema de Gestión de Base de Datos (SGBDR). Las BD más tradicionales dedicaban mucho tiempo en el rendimiento, mientras que las NoSQL están altamente optimizadas para las operaciones de recuperación y agregación. Hoy en día el uso de este tipo de repositorios permite a los desarrolladores trabajar con aplicaciones que mueven una gran cantidad de volúmenes de datos y con una evolución constante [20].

Un ejemplo de este tipo de almacenes de datos es MongoDB, un sistema de bases de datos NoSQL, orientado a documentos y de código abierto. Dicho sistema, es que ha sido seleccionado para realizar este TFG, por su alta compatibilidad e interoperabilidad con los requisitos impuestos por la empresa.

La tabla siguiente proviene del sitio web de MongoDB [22]:

	BD SQL	BD NoSQL
Historia	Desarrolladas a finales de 1970 para respaldar las aplicaciones con almacenamiento de datos	Desarrolladas a finales de 2000 con el objetivo de superar las limitaciones de las BD SQL
Tipos	Un tipo con pequeñas variaciones	Clave-valor, documentales, grafos
Ejemplos	Microsoft SQL	MongoDB
Escalabilidad	Escalado vertical, necesario potenciar el servidor en caso de aumentar la demanda. En el proceso de repartición de las BD se suele perder funcionalidades como las operaciones JOIN.	Escalado horizontal, en caso de querer aumentar la potencia, el administrador solo debe añadir servidores básicos o instancias en la nube. La propia BD distribuye los datos en los servidores.
Manipulación de datos	Instrucciones Select, Insert, Update	A través de APIs

Tabla 2: Diferencias entre SGBD y NoSQL

Los bloques de datos han sido descargados en un fichero JSON (en inglés JavaScript Object Notation), que posteriormente se ha almacenado en MongoDB. Aunque en MongoDB se puede introducir directamente un documento JSON, tal y como se descarga del sitio web de AEMET, el documento proporcionado por la API de AEMET viene con todos los datos en formato string como se puede ver en la figura 15. Es por eso que hay que tratar cada campo de los datos obtenidos y almacenarlos en el formato adecuado. En el punto 3.6.2 AEMET OpenData se ha comentado que existen dos enlaces. A continuación, se muestra una figura que hace referencia a un día en concreto que ha sido solicitado a la API. Como se puede observar, los campos están en formato string.

```
[ {
  "fecha" : "2021-01-01",
  "indicativo" : "9434",
  "nombre" : "ZARAGOZA AEROPUERTO",
  "provincia" : "ZARAGOZA",
  "altitud" : "249",
  "tmed" : "5,8",
  "prec" : "1,2",
  "tmin" : "2,4",
  "horatmin" : "23:59",
  "tmax" : "9,3",
  "horatmax" : "15:20",
  "dir" : "31",
  "velmedia" : "6,9",
  "racha" : "12,8",
  "horaracha" : "12:30",
  "sol" : "6,4",
  "presMax" : "978,8",
  "horaPresMax" : "23",
  "presMin" : "976,2",
  "horaPresMin" : "14"
}, {
```


Figura 15: Datos descargados de la API de AEMET en formato JSON

Llegado este punto, se va a introducir el Business Intelligence (BI) que hace referencia al uso que hacen las empresas de los datos y de la información que tienen (aplicaciones, productos, tecnología, competidores, mercado, clientes, proveedores o incluso de los empleados que tienen).

Mediante las herramientas y técnicas de Extracción, Transformación y Carga (ETC en castellano o ETL en inglés) se extraen los datos de diversas fuentes, se refinan y se guardan a través de consultas a los almacenes de datos.

En este TFG se ha empleado AEMET como fuente de información ya que es lo suficientemente fiable como para poder crear una RNA y que trabaje más o menos bien, por lo que es muy importante el origen de los datos, el preprocesado, normalización y postprocesado de los mismos.

Antes de introducir los datos a la BD, se necesita cambiar el formato de los campos para que se pueda trabajar con ellos con normalidad y además de no entorpecer y ralentizar el trabajo, a continuación, se muestra un día almacenado en la BD con los tipos de campos correspondientes.

```
_id: ObjectId("6096cf3fe387a4d7e7c123e9")
fecha: 1968-01-12T00:00:00.000+00:00
indicativo: "9434"
nombre: "ZARAGOZA, AEROPUERTO"
provincia: "ZARAGOZA"
altitud: 249
tmed: 8.3
prec: 0
tmin: 4.1
horatmin: "01:00"
tmax: 12.5
horatmax: "16:00"
dir: 29
velmedia: 11.4
racha: 20
horaracha: "16:19"
sol: 5.3
presMax: 991.4
horaPresMax: "21"
presMin: 12.5
horaPresMin: "15"
```


Figura 16: Datos almacenados en la BD de MongoDB con sus respectivos formatos

Una vez se tienen introducidos los datos en la BD, se hacen bloques de esos datos tanto para el entrenamiento como para la validación de la RNA. En un primer momento se utilizó una sola estación meteorológica, más en concreto la estación de Zaragoza Aeropuerto, con la que se ha creado la RNA, entrenado y validado ese entrenamiento. Tras haber comprobado la correcta creación y funcionamiento de la RNA, se han añadido 3 estaciones meteorológicas más, es decir, la RNA final va a hacer uso de datos de 4 estaciones que son Zaragoza Aeropuerto, Logroño, Valencia y Huesca. Podrían haber sido otras, pero lo que interesa es que coincidan en el número de datos totales que se pueden adquirir de cada una de ellas.

Ya que la temperatura de cualquier lugar está condicionada, por la climatología que hace en los lugares que lo rodean, lo ideal es que el punto en el que se quiere predecir la temperatura esté rodeado de otras estaciones de las cuales se puedan obtener los datos almacenados. AEMET no dispone de datos desde el año 1968 de todos los lugares en los que tiene una estación meteorológica, ya sea porque se instalaron más tarde o porque falló en su momento la recogida de datos. Esta es la razón por la que se han utilizado las localizaciones que se puede observar en la siguiente figura, siendo Zaragoza el lugar en el que se va a realizar la previsión.

Figura 17: Mapa de la península Ibérica

El siguiente paso es el preprocesado o limpieza de datos. Aquellos días en los que la estación por el motivo que sea, no ha podido registrar algún campo, esa fila de datos se desecha ya que se desconoce parte de los datos y por lo tanto puede repercutir en el entrenamiento de la RNA. Aquellas filas donde existen valores no numéricos registrados como NaN (en inglés Not a Number) entre los datos de las estaciones, han sido eliminadas. Lo más probable es que se acabe utilizando un menor número de datos de los que se han obtenido al principio.

A continuación, se normalizan los datos, ya que son datos numéricos y deben de estar comprendidos entre (0, 1) o (-1, 1). Para llevar a cabo esta tarea se ha utilizado la función MinMaxScaler de Sklearn. Después hay que crear los bloques de datos como se ha mencionado y repartir esos bloques en: bloques de entrenamiento (80% de los datos) utilizados, para llevar a cabo el entrenamiento y, bloques de validación (20% de los datos) empleados para validar la RNA creada.

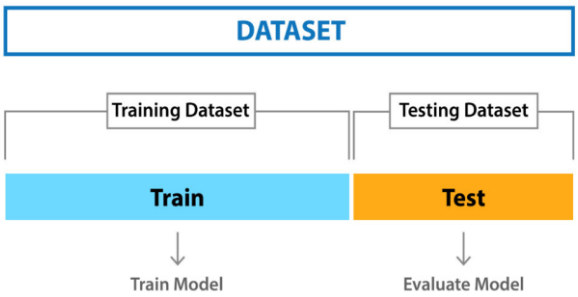

Figura 18: Estructura del dataset dividida en bloques de entrenamiento y test

Al crear una RNA que trata con series temporales, los bloques de datos que se han creado han sido de 5 días más 1 día, es decir, si la fecha inicial comienza en el día 1 de enero de 1968, el

primer bloque creado va a contener hasta el día 5 de enero de 1968 como datos de “pasado” y el día 6 de enero de 1968 es el día de “futuro”. El día de futuro es consecuencia de los cinco días anteriores. El segundo bloque empieza en el día 2 de enero de 1968 y acaba en el día 7 de enero de 1968, siendo este último, el día “futuro”, de esta manera es como se han creado los diferentes bloques hasta llegar al último día. Estos parámetros pueden ser modificados ya que, en un principio, se trabajó con bloques de 5+1, pero después de haber realizado una serie de comprobaciones, se ha llegado a la conclusión de que la RNA con 7+1 da un mejor resultado.

El tercer paso que se debe seguir para poder crear nuestra RNA es la creación de la arquitectura [23][24][25][26][27]. Para la que se ha creado un modelo de tipo “Sequential” donde las capas de neuronas van a ser secuenciales, es decir, una detrás de otra.

Anteriormente se ha comentado que se han creado diferentes RNA con diferentes configuraciones y se ha comprobado cuál es la más óptima (esta parte se explica en el punto 6 de este mismo documento). Se han usado diferentes lotes de números de datos y también se ha utilizado el optimizador de hiperparámetros llamado Keras Tuner, para optimizar la cantidad de neuronas que se van a emplear. Tanto si se ha utilizado optimizador como si no, todas las redes creadas tienen 3 capas, una capa de entrada, una capa oculta LSTM y una capa de salida. Es en la cantidad de nodos empleados en la capa oculta y el tamaño del bloque donde se encuentra la diferencia entre cada RNA. Para la creación, sin hacer uso del optimizador, la cantidad de neuronas ha sido de ⅔ del total de datos obtenidos, mientras que con el optimizador de hiperparámetros ha sido calculado automáticamente. A continuación, se muestra el código empleado para la creación de la RNA sin optimización, con Keras Tuner y el código de entrenamiento.

Código de creación de RNA sin optimización:

```
def create(self):
    '''
    Función que crea la red neuronal del tipo LSTM (Long Short-Term
Memory), entrena y predice
    '''
    percentage = 2/3
    num_nodes = int(self.input_size[0] * self.input_size[1] *
percentage) + 1 # Se suma 1 por si es decimal
    inputlayer = keras.layers.Input(shape=(self.input_size[0],
self.input_size[1]))
    lstm = keras.layers.LSTM(num_nodes,
activation='relu')(inputlayer)
    outputlayer = keras.layers.Dense(1)(lstm)
    model = keras.Model(inputs=inputlayer, outputs=outputlayer)

    model.compile(optimizer='adam', loss='mean_squared_error')
    plot_model(model, 'net.png', show_shapes=True)
    self.model = model
```

Figura 19: Código de creación de RNA sin optimización

Código de creación de RNA con Keras Tuner:

```
def create_tuner(self, hp):
    '''
    Función que crea la red neuronal del tipo LSTM (Long Short-Term
Memory), entrena y predice, en este caso,
    utiliza Keras Tuner para optimizar la RNA
    '''
    inputlayer = keras.layers.Input(shape=(self.input_size[0],
self.input_size[1]))
    lstm = keras.layers.LSTM(units=hp.Int('units', min_value=20,
max_value=80, step=5), activation='relu')(inputlayer)
    outputlayer = keras.layers.Dense(1)(lstm)
    model = keras.Model(inputs=inputlayer, outputs=outputlayer)

    # Ratio de optimización para keras tuner
    # Elige un valor optimo entre los siguientes 0.01, 0.001, o
0.0001

model.compile(optimizer=keras.optimizers.Adam(hp.Choice('learning_ra
te', values=[1e-2, 1e-3, 1e-4])),
                  loss='mean_squared_error')
    plot_model(model, 'net.png', show_shapes=True)
    self.model = model
    return model
```

Figura 20: Código de creación de RNA con Keras Tuner

Código de entrenamiento:

```
def train(self):
    '''
    Función que entrena el modelo de red neuronal creado
    '''
    if not self.model:
        raise ValueError('A Neural Network has to be created first')
    else:
        history = self.model.fit(self.ds_train, epochs=20, verbose=1,
validation_data=(self.ds_val))
        NeuralNetwork.show_loss(history, "Training and Validation
Loss")
        return history
```

Figura 21: Código de entrenamiento de RNA

Las diferentes RNA creadas han sido guardadas con formato *h5* para evitar el entrenamiento cada vez que se quiera ejecutar el programa, de esta forma, tan solo cargando la red neuronal deseada se puede comenzar a predecir.

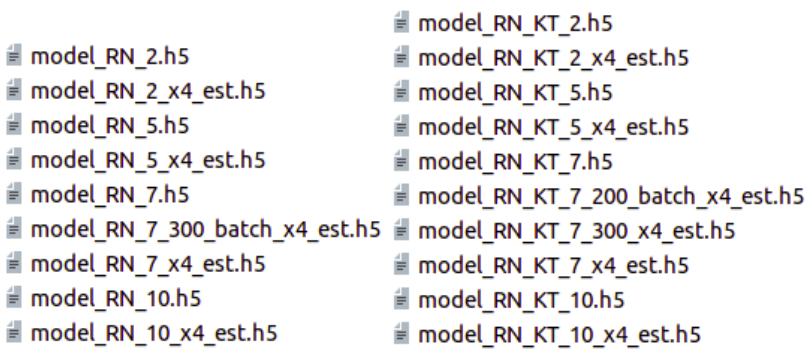

Figura 22: Modelos de RNAs creados y guardados

El cuarto paso es compilar la red neuronal, que transformará la secuencia de capas en una serie de matrices altamente eficientes. Para ello, el algoritmo de optimización a utilizar para el proceso de entrenamiento y la función de pérdida utilizada para evaluar la RNA, que es minimizada por el algoritmo de optimización, requiere que se especifiquen una serie de parámetros. En este TFG se han empleado el algoritmo de optimización “adam” y la función de pérdida de error cuadrático medio “mean_squared_error”. También se pueden especificar las métricas que desea recopilar al ajustar el modelo, además de la función de pérdida (accuracy, binaryAccuracy, RootMeanSquaredError, etc).

El siguiente paso es ajustar el modelo, lo que significa adaptar los pesos a un conjunto de datos de formación. El algoritmo requiere que la RNA sea entrenada para un número específico de épocas (epochs) al conjunto de datos de entrenamiento. Cada época puede dividirse en grupos llamados lotes (batch size).

Se han definido en 20 las epochs y el tamaño de datos que tomará cada lote o batch en 10. Pero tras las pruebas realizadas, se ha comprobado que es mejor la RNA que hace uso de lotes de tamaño 200.

Tras tener la RNA creada, entrenada y guardada se han realizado una serie de pruebas. Comparar las diferentes redes que se han creado, se trata del último paso: la evaluación de la RNA.

Con el histórico de datos se ha predicho la temperatura media y comparado con la temperatura media para esos días. Las temperaturas predichas se han almacenado en una nueva colección en MongoDB para poder realizar los diferentes gráficos.

La figura 23 muestra las predicciones de cada modelo de RNA, empleadas para la creación del diagrama de Taylor.

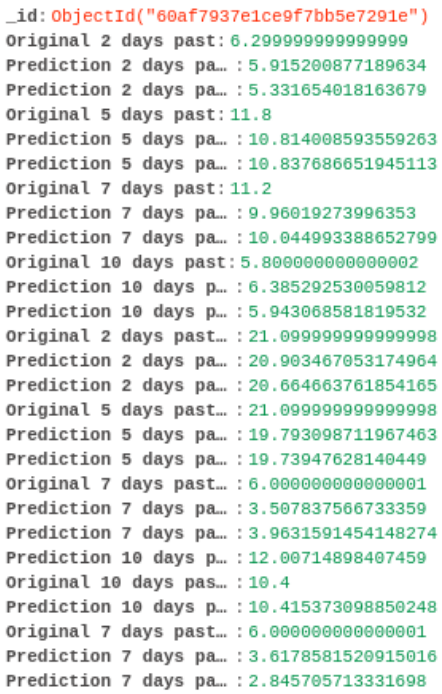

Figura 23: Predicciones de cada modelo RNA

3.8 Alojamiento del proyecto (Gitlab UNIZAR)

El software desarrollado para este TFG ha sido alojado en el repositorio de Gitlab de la Universidad de Zaragoza (UNIZAR) con el propósito de que pueda ser utilizado por cualquier persona, cumpliendo con las premisas de la licencia software seleccionada.

El enlace del proyecto es el siguiente:

https://gitlab.unizar.es/701945/ai-meteo.git

4. Evaluación de RNA

En este punto del TFG se van a mostrar los resultados obtenidos y se van a comparar los diferentes modelos de RNA creados para su evaluación. Se va a dividir la sección en dos subapartados con el objetivo de distinguir los modelos creados con los datos recogidos de una sola estación meteorológica, de aquellos modelos en los que se han utilizado cuatro estaciones meteorológicas. En los modelos sin optimización se ha utilizado el ⅔ del total de datos para el número de nodos. En el caso de los modelos creados con Keras Tuner, han sido calculados, de forma automática, comprobando qué cantidad de nodos proporciona una menor pérdida.

Destacar antes de entrar en detalle en cada modelo, que los modelos han sido creados con los parámetros por defecto y posteriormente, se han creado modelos optimizados haciendo uso de Keras Tuner, optimizando los hiperparámetros de manera automática. El mejor modelo obtenido ha sido uno de los implementados con Keras Tuner.

4.1 Estructura de la RNA

En cuanto a la estructura de los modelos de RNA creados con 1 sola estación meteorológica, la capa de entrada tiene 10 dimensiones, 1 capa oculta del tipo LSTM y 1 capa de salida de 1 dimensión. Por otro lado, los modelos creados con 4 estaciones tienen en la capa de entrada 40 dimensiones, ya que se va a hacer uso de muchos más datos provenientes de más estaciones, 1 capa oculta del tipo LSTM y una capa de salida de 1 dimensión, ya que el propósito es predecir sólo una temperatura media. Es en la capa oculta donde se encuentra la mayor diferencia.

En primer lugar, se puede observar la estructura del modelo de 4 estaciones meteorológicas con 7+1 días y tamaño de lote de 10. En la segunda figura, se muestra la estructura del modelo de 4 estaciones con 7+1 días y un tamaño de lote de 200 y, además, optimizado por Keras Tuner. Se han elegido estos dos modelos por ser los mejores, en los siguientes puntos se entrará más en detalle del porqué de la elección de estos.

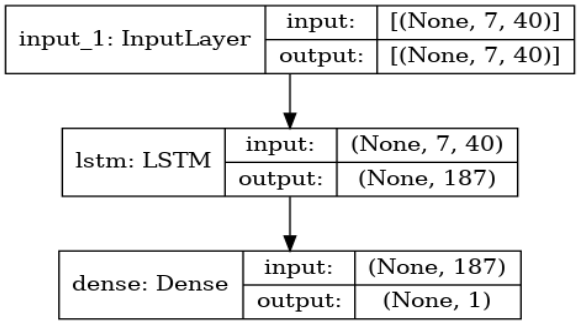

Figura 24: Estructura RNA 7+1 días sin optimización

Como se puede apreciar si se compara la figura anterior con la siguiente figura, la diferencia se encuentra en la capa oculta. Se puede verificar que la capa de entrada es de 40 dimensiones. En la capa oculta se puede ver claramente que en el primer caso se ha creado el modelo con 187 nodos y en el segundo caso con 45 nodos es suficiente.

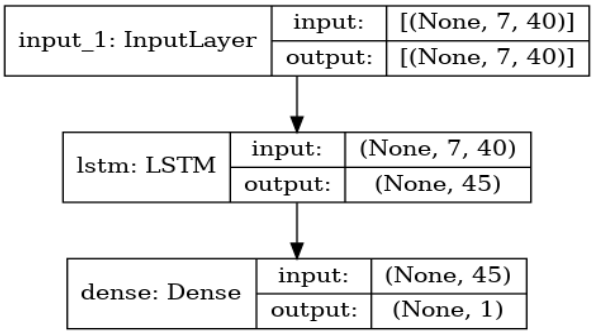

Figura 25: Estructura RNA 7+1 días optimizada

4.2 Entrenamiento de la RNA

Cuando se crea una RNA es muy importante hacer un seguimiento del entrenamiento que está recibiendo cada modelo, para poder comprobar si ha habido algún problema o en algún momento se ha producido alguna pérdida.

Tal y como se ha hecho en el punto anterior y como se va a hacer en los puntos posteriores, sólo se van a comentar los resultados de los mejores modelos.

```
Epoch 9/20
435/435 [==============================] - 4s 9ms/step - loss: 0.0031 - val_loss: 0.0029
Epoch 10/20
435/435 [==============================] - 4s 9ms/step - loss: 0.0031 - val_loss: 0.0029
Epoch 11/20
435/435 [==============================] - 4s 9ms/step - loss: 0.0030 - val_loss: 0.0029
Epoch 12/20
435/435 [==============================] - 4s 9ms/step - loss: 0.0029 - val_loss: 0.0028
Epoch 13/20
435/435 [==============================] - 4s 9ms/step - loss: 0.0029 - val_loss: 0.0028
Epoch 14/20
435/435 [==============================] - 4s 9ms/step - loss: 0.0028 - val_loss: 0.0028
Epoch 15/20
435/435 [==============================] - 4s 9ms/step - loss: 0.0027 - val_loss: 0.0027
Epoch 16/20
435/435 [==============================] - 4s 9ms/step - loss: 0.0027 - val_loss: 0.0027
Epoch 17/20
435/435 [==============================] - 4s 9ms/step - loss: 0.0026 - val_loss: 0.0027
Epoch 18/20
435/435 [==============================] - 4s 9ms/step - loss: 0.0026 - val_loss: 0.0027
Epoch 19/20
435/435 [==============================] - 4s 9ms/step - loss: 0.0025 - val_loss: 0.0027
Epoch 20/20
435/435 [==============================] - 4s 9ms/step - loss: 0.0025 - val_loss: 0.0027
```

Figura 26: Entrenamiento del modelo sin optimización

La figura 26 muestra el entrenamiento de la RNA de 7+1 días con 1 estación meteorológica sin optimización.

En la figura 27 se muestra cómo ha sido la pérdida del modelo, en la que la curva azul hace referencia a la pérdida del entrenamiento y la curva roja a la pérdida de la validación.

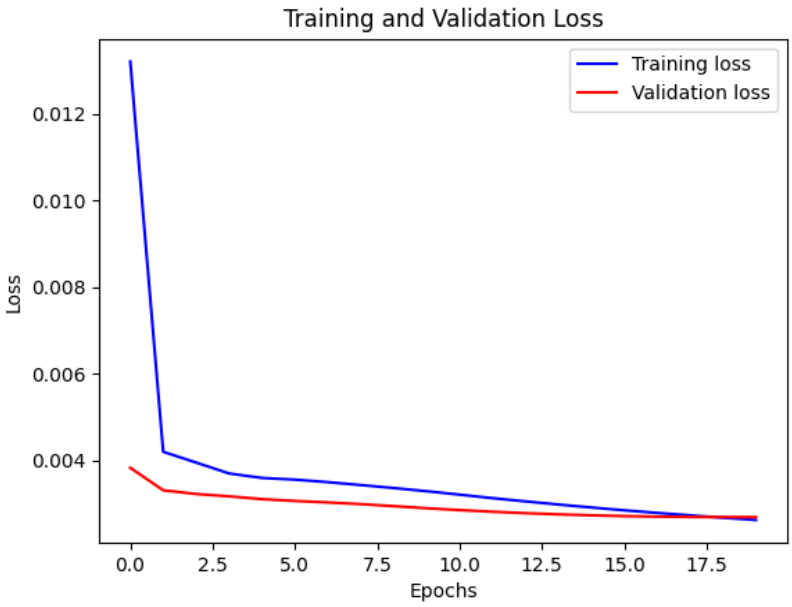

Figura 27: Gráfica de pérdida del modelo sin optimización

Por otro lado, si se compara el mejor modelo con tamaño de lote de 200, ambas figuras van a ser completamente diferentes por el hecho de ser diferentes modelos.

```
Trial 10 Complete [00h 00m 15s]
val_loss: 0.02096428396180272

Best val_loss So Far: 0.004253384470939636
Total elapsed time: 00h 02m 32s
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 7, 40)]           0
_________________________________________________________________
lstm (LSTM)                  (None, 45)                15480
_________________________________________________________________
dense (Dense)                (None, 1)                 46
=================================================================
Total params: 15,526
Trainable params: 15,526
Non-trainable params: 0
```

Figura 28: Entrenamiento del modelo con optimización

Se ha empleado una configuración totalmente diferente, en el modelo optimizado el tamaño de bloque es de 200 y 20 epochs mientras que en el modelo sin optimización el tamaño de bloque es de 10 y la misma cantidad de epochs.

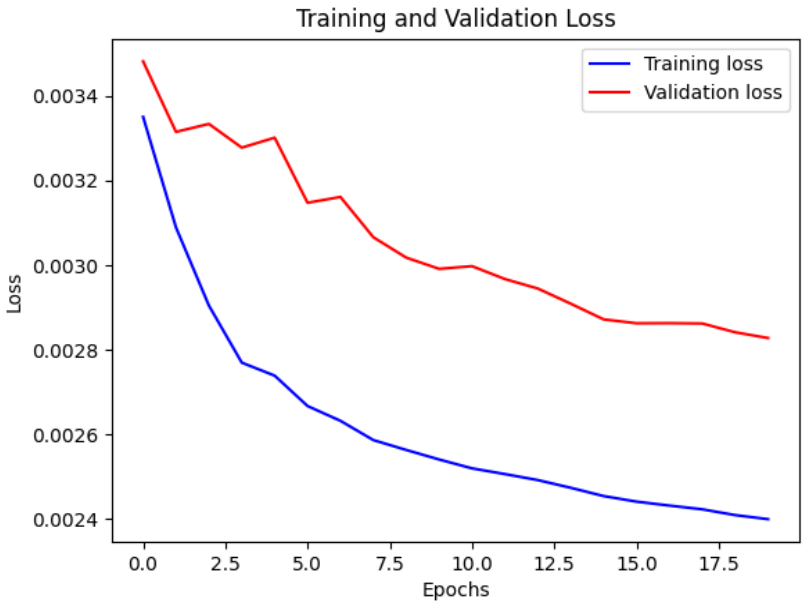

Figura 29: Gráfica de pérdida del modelo con optimización

La figura 29 correspondiente a la gráfica de pérdida que no es tan "bonita" como la figura 27 pero como se verá en los puntos siguientes no se puede afirmar que una pueda ser mejor que otra, si la evaluación dependiera sólo de este tipo de gráficas.

4.3 Comparación de datos reales y datos predichos

Tras haber creado y entrenado los diferentes modelos y como objetivo de este TFG, se han creado una serie de gráficas y figuras, en las que se pueden observar las temperaturas medias predichas comparadas con las que en realidad deberían ser.

Mencionar que al haber tenido que eliminar bloques de datos por el hecho de existir campos desconocidos, para no perturbar los modelos y, por tanto, ser lo más realistas posibles al no haber inventado los datos faltantes, las temperaturas predichas pueden variar y no ajustarse al 100% con la real.

La figura 30 hace referencia a las temperaturas medias de la estación de Zaragoza Aeropuerto, los símbolos rojos simulan la temperatura real en ese lugar y los azules son las temperaturas medias que el modelo ha predicho.

Una manera de comprobar que se están empleando los datos correctamente es viendo en la gráfica que, más o menos, cada 365 días empieza un nuevo año. Además, también se puede ver como las temperaturas oscilan entre temperaturas bajas y temperaturas altas, haciendo referencia a las diferentes estaciones del año.

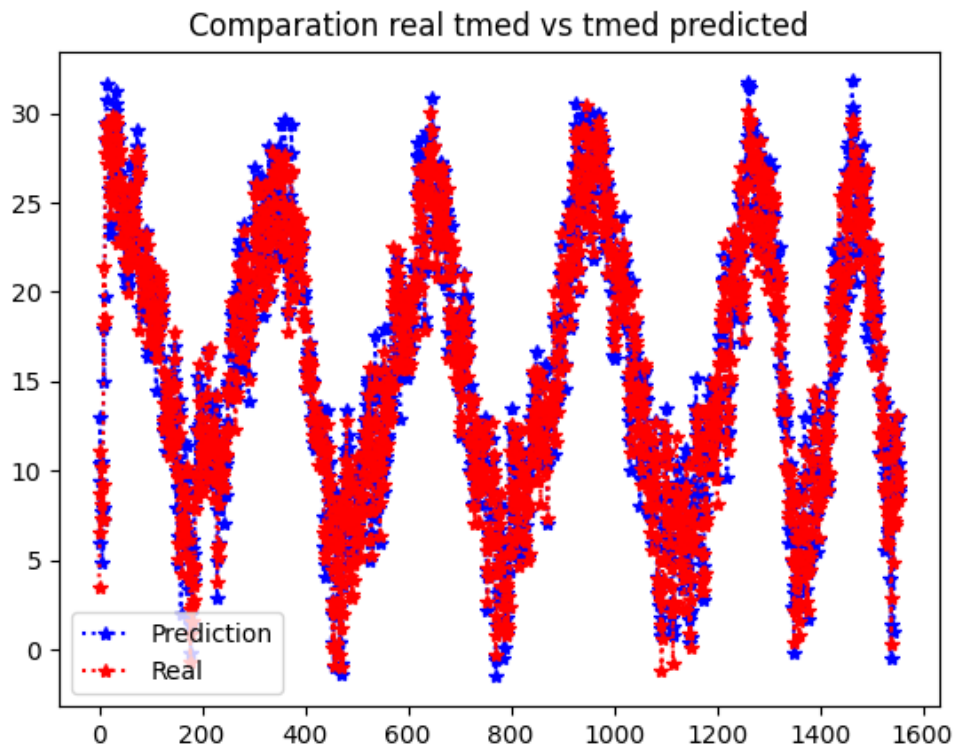

Figura 30: Comparación temperatura media predicha y real del modelo sin optimizar

La figura 31 muestra una comparación entre los datos reales y los datos predichos del modelo sin optimizar.

	Original 7 days past final	Prediction 7 days past final
0	6.0	3.507838
1	9.4	6.578784
2	13.0	8.735444
3	10.4	11.122604
4	10.5	8.183269
...	...	...
1551	12.3	11.222324
1552	8.8	9.165071
1553	9.0	8.351257
1554	10.4	9.986435
1555	13.0	9.498288

[1556 rows x 2 columns]

Figura 31: Comparación de datos predichos y reales del modelo sin optimizar

Observando la figura 32 perteneciente al modelo optimizado, se ve una pequeña diferencia en la muestra. Las temperaturas más altas predichas destacan menos que en el modelo anterior.

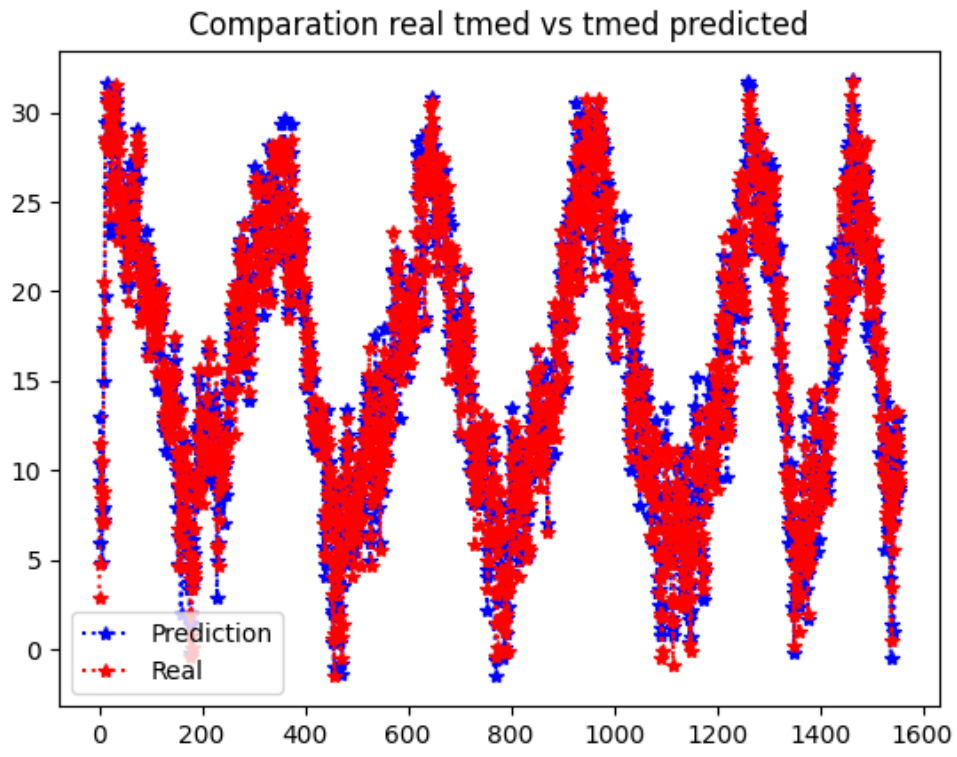

Figura 32: Comparación temperatura media predicha y real del modelo optimizado

Como en el ejemplo anterior, en la siguiente figura se hace una comparación de los datos reales y predichos. Estas son las temperaturas con las que se han obtenido las gráficas de comparación.

	Original 7 days past final 200 batch	Prediction 7 days past KT final 200 batch
0	6.0	2.871220
1	9.4	4.750340
2	13.0	9.034241
3	10.4	11.496919
4	10.5	8.405339
...	...	...
1551	12.3	11.950413
1552	8.8	11.069609
1553	9.0	9.484067
1554	10.4	9.205780
1555	13.0	9.852963

[1556 rows x 2 columns]

Figura 33: Comparación de datos predichos y reales del modelo optimizado

Sólo llevando a cabo una comparación de las medias, no es suficiente para evaluar si funciona dentro de lo esperado o correctamente. Es por ello, que se hace necesario el gráfico Q-Q.

4.4 Gráfico Q-Q (Cuantil-Cuantil)

Un gráfico Cuantil-Cuantil es una representación gráfica que permite observar cuán cerca está la distribución de un conjunto de datos a alguna distribución ideal o comparar la distribución de dos conjuntos de datos. La idea de este tipo de gráficos es que, si las dos distribuciones coinciden, veremos un gráfico muy parecido a una recta. Tanto más parecido cuanto mayor sea la coincidencia.

Para hablar con profesionalidad, se van a utilizar los términos de sobre-estima y sub-estima. Sobre-estima se refiere a cuando la distribución de datos se encuentra por encima de la distribución ideal, mientras que, por el contrario, sub-estima es cuando la distribución de datos se halla en la parte inferior de la distribución ideal.

Sabiendo cómo se debe interpretar este tipo de gráficos, se van a comparar los gráficos de los mejores modelos de RNA.

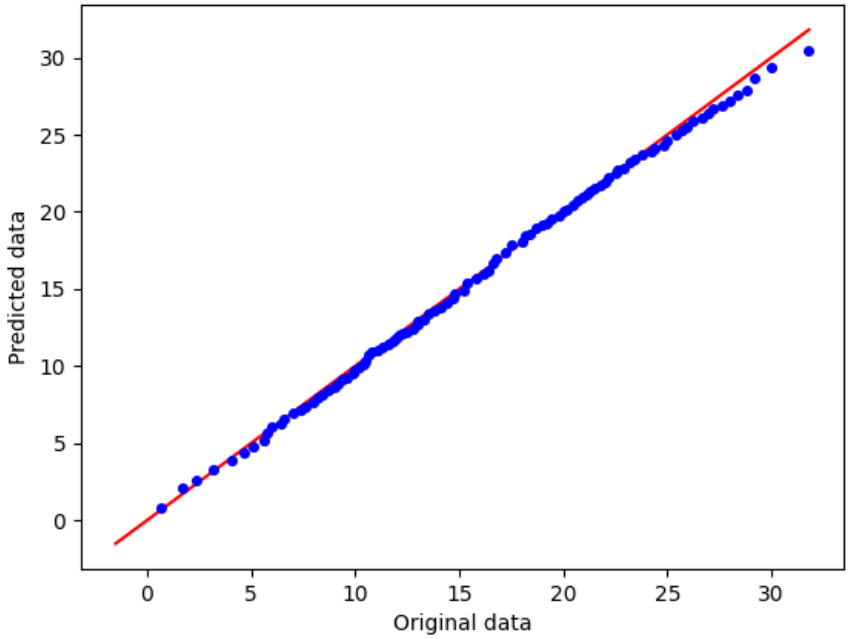

Figura 34: Gráfica Q-Q del modelo sin optimizar

El modelo anterior sub-estima en la cola superior (no es capaz de predecir bien con temperaturas muy altas).

En el segundo caso, si bien sobre-estima en la cola inferior (no es capaz de predecir bien temperaturas muy bajas) lo hace de una forma más sutil que el ejemplo anterior. Además, se puede ver cómo se adapta mejor a la distribución ideal.

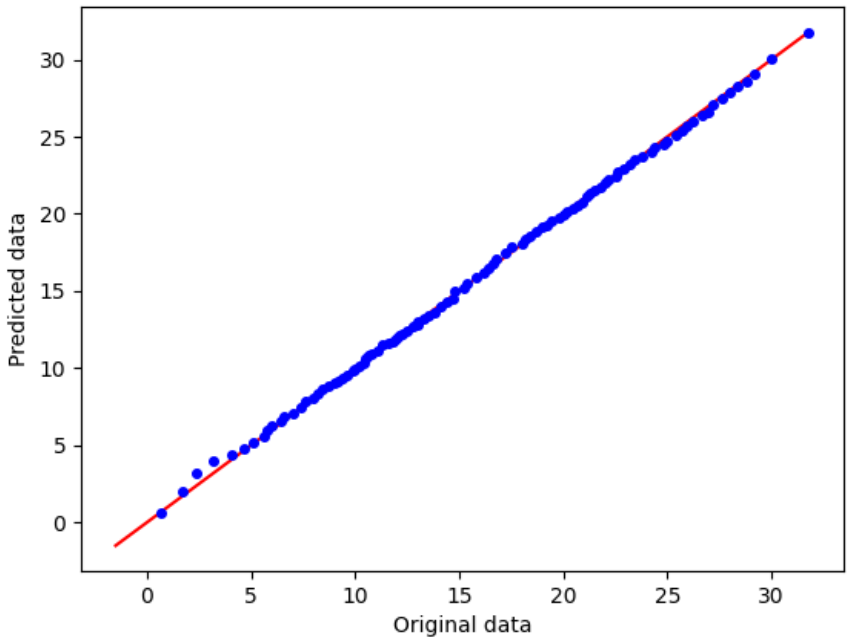

Figura 35: Gráfica Q-Q del modelo optimizado

El siguiente código es el implementado para la creación de las gráficas Q-Q.

```
def show_qqplot(y_orig, y_pred, quantiles=np.linspace(0.01, 1,
100)):
    '''
    Función que crea una gráfica en la que se muestra los cuartiles
de
    los datos reales y los cuartiles de los datos predichos.
    :param y_orig: dataframe de los datos reales
    :param y_pred: dataframe de los datos predichos
    :param quantiles: cuantiles en lo que se van a dividir los datos
    '''
    fulldata = np.vstack((y_orig, y_pred)).flatten()
    minval = np.min(fulldata)
    maxval = np.max(fulldata)

    q_orig = mquantiles(y_orig, quantiles)
    q_pred = mquantiles(y_pred, quantiles)

    plt.plot([minval, maxval], [minval, maxval], 'r')
    plt.plot(q_orig, q_pred, 'o', markersize=4, color='blue')
    plt.xlabel('Original data')
    plt.ylabel('Predicted data')
    plt.show()
```

Figura 36: Código para la creación de gráficas Q-Q

Con los datos obtenidos en este subapartado, se puede ver que se observan mejoras con respecto a la distribución ideal, llegando a poder tomar algunas decisiones. Con todo y con eso, se va a probar a través del diagrama de Taylor.

4.5 Diagrama de Taylor

El diagrama de Taylor proporciona un resumen estadístico conciso de que también los patrones coinciden entre sí en términos de su correlación, su diferencia de raíz cuadrada media y la razón de sus varianzas. Facilita un marco visual para comparar un conjunto de variables de uno o más conjuntos de datos de prueba con uno o más conjuntos de datos de referencia. Por lo general, los conjuntos de datos de prueba son experimentos de modelo. Se indica el punto de referencia observado donde la correlación es 1, color azul, y el Error de la Raíz Cuadrada Media (RMSE o RMSD), de color verde, es 0. Si el punto de simulación del modelo está cerca del punto observado de color rojo, significa que son similares en términos de desviación estándar, su correlación es alta y su RMSE es cercano a cero.

Para un mejor entendimiento, la desviación estándar es una de las medidas de dispersiones más comunes, indica cómo de dispersos están los datos con respecto a la media. Mientras mayor sea la desviación estándar, mayor será la dispersión de los datos.

Por otro lado, la RMSE no es nada más que la raíz del Error Cuadrático Medio (MSE), donde $\hat{y}_i$ es el valor estimado e y_i es el valor real. MSE calcula la diferencia cuadrada para cada punto de las predicciones y el objetivo, luego hace un promedio de los valores.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

Figura 37: Fórmula del RMSE

El coeficiente de correlación es una medida que permite conocer el grado de asociación lineal entre dos variables. Tiene la fórmula matemática que se puede ver a continuación, siendo el numerador la covarianza y el denominador la desviación estándar de X por la desviación estándar de Y.

$$r = \frac{S_{XY}}{S_X S_Y}$$

Figura 38: Fórmula del coeficiente de correlación

Una correlación más alta muestra un nivel más alto de concordancia entre los datos observados y simulados. La correlación desciende a medida que un modelo se mueve hacia sectores superiores en el gráfico. Por esta razón, se deben tener en cuenta también los gráficos Q-Q ya obtenidos evaluar los modelos con todos los gráficos y diagramas disponibles y no decantarse por un modelo u otro solo teniendo en cuenta uno de ellos.

Se pasa a presentar el diagrama de Taylor obtenido con todos los modelos de RNA creados. A cada modelo se le ha dado un nombre compuesto por una “M” y un número siguiendo el orden de creación. Por lo tanto, la disposición de los nombres quedaría de la siguiente manera:

	RNA sin optimizar	RNA optimizada (Keras Tuner)
2 datos de pasado	M1	M5
5 datos de pasado	M2	M6
7 datos de pasado	M3	M7
10 datos de pasado	M4	M8
2 datos de pasado x4 estaciones	M9	M13
5 datos de pasado x4 estaciones	M10	M14
7 datos de pasado x4 estaciones	M11	M15
10 datos de pasado x4 estaciones	M12	M16
7 datos de pasado x4 estaciones y 300 batch size	M17	M18
7 datos de pasado x4 estaciones y 200 batch size	No existe este modelo	M19

Tabla 3: Diferentes modelos RNA creados

Tras haber creado el diagrama de Taylor, con todos los modelos y métricas ya comentadas, se ha creado un solo diagrama con todos los modelos y sus respectivos nombres. Se podría haber creado un diagrama por cada RNA, pero la idea de este tipo de diagramas es hacer una comparación con todos los modelos de RNA creados, permitiendo así una mejor visualización de las posiciones de cada uno de ellos.

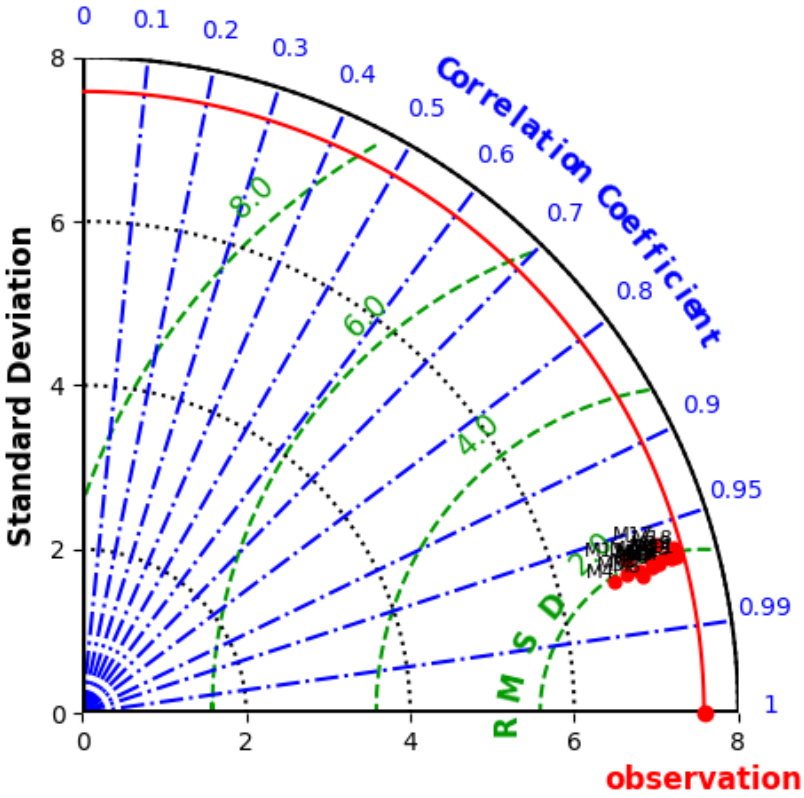

Figura 39: Diagrama de Taylor

Perfectamente se pueden visualizar todos los modelos en una zona muy concreta del diagrama. Además, se puede destacar que todos ellos están bastante bien situados, debido a que se encuentran en una zona relativamente cercana al punto de observación. La imagen siguiente se trata de un zoom, en la localización de los modelos, que permite mejorar la visualización de cada uno de ellos y ver cuál es el mejor situado.

En el zoom realizado al diagrama de Taylor se pueden ver todos los modelos dispersos en él. El modelo M4, aun sabiendo que el mejor modelo es el que se encuentra más cercano al punto de observación y situado lo más abajo posible, es el peor de todos los modelos creados. Por el contrario, tanto el modelo M11 como el M19 se encuentran muy cercanos a la curva de observación.

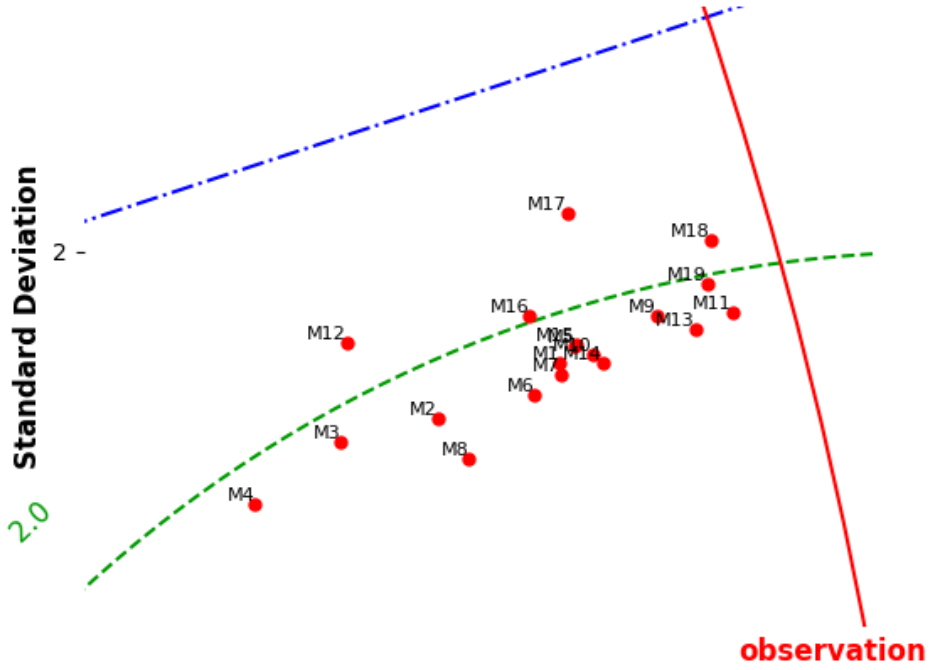

Figura 40: Zoom en el diagrama de Taylor

Teniendo en cuenta sólo este diagrama, el modelo M11 es el mejor de todos los creados, mientras que M19 se encuentra en una situación muy similar y como ya se ha visto en el diagrama Q-Q consigue adaptarse y parecerse más a la distribución ideal.

En cuanto a la realización de este diagrama, hablando de programación, se ha tenido que instalar desde el repositorio de GitHub el paquete SkillMetrics de Peter Rochford [28].

Puede ser instalado con la siguiente instrucción en Linux:

```
$ pip install SkillMetrics
```

El siguiente código ha sido implementado para la creación del diagrama de Taylor:

```
def show_taylor_plot(df):
    '''
    Función que crea el diagrama de Taylor
```

```
    :param df_orig: dataframe de los datos reales
    :param df_pred: dataframe de los datos predichos
    '''
    taylor_stats1 = sm.taylor_statistics(df["Prediction 2 days
past"].squeeze(), df["Original 2 days past"].squeeze(),
                                         'Prediction')
    taylor_stats2 = sm.taylor_statistics(df["Prediction 5 days
past"].squeeze(), df["Original 5 days past"].squeeze(),
                                         'Prediction')
    taylor_stats3 = sm.taylor_statistics(df["Prediction 7 days
past"].squeeze(), df["Original 7 days past"].squeeze(),
                                         'Prediction')
    taylor_stats4 = sm.taylor_statistics(df["Prediction 10 days
past"].squeeze(), df["Original 10 days past"].squeeze(),
                                         'Prediction')
    taylor_stats5 = sm.taylor_statistics(df["Prediction 2 days past
KT"].squeeze(), df["Original 2 days past"].squeeze(),
                                         'Prediction')
    taylor_stats6 = sm.taylor_statistics(df["Prediction 5 days past
KT"].squeeze(), df["Original 5 days past"].squeeze(),
                                         'Prediction')
    taylor_stats7 = sm.taylor_statistics(df["Prediction 7 days past
KT"].squeeze(), df["Original 7 days past"].squeeze(),
                                         'Prediction')
    taylor_stats8 = sm.taylor_statistics(df["Prediction 10 days past
KT"].squeeze(), df["Original 10 days past"].squeeze(),
                                         'Prediction')
    taylor_stats9 = sm.taylor_statistics(df["Prediction 2 days past
final"].squeeze(), df["Original 2 days past final"].squeeze(),
                                         'Prediction')
    taylor_stats10 = sm.taylor_statistics(df["Prediction 5 days past
final"].squeeze(), df["Original 5 days past final"].squeeze(),
                                          'Prediction')
    taylor_stats11 = sm.taylor_statistics(df["Prediction 7 days past
final"].squeeze(), df["Original 7 days past final"].squeeze(),
                                         'Prediction')
    taylor_stats12 = sm.taylor_statistics(df["Prediction 10 days past
final"].squeeze(), df["Original 10 days past final"].squeeze(),
                                         'Prediction')
    taylor_stats13 = sm.taylor_statistics(df["Prediction 2 days past
KT final"].squeeze(), df["Original 2 days past final"].squeeze(),
                                          'Prediction')
    taylor_stats14 = sm.taylor_statistics(df["Prediction 5 days past
KT final"].squeeze(), df["Original 5 days past final"].squeeze(),
                                         'Prediction')
    taylor_stats15 = sm.taylor_statistics(df["Prediction 7 days past
KT final"].squeeze(), df["Original 7 days past final"].squeeze(),
                                         'Prediction')
    taylor_stats16 = sm.taylor_statistics(df["Prediction 10 days past
KT final"].squeeze(), df["Original 10 days past final"].squeeze(),
                                          'Prediction')
    taylor_stats17 = sm.taylor_statistics(df["Prediction 7 days past
final 300 batch"].squeeze(), df["Original 7 days past final 300
batch"].squeeze(),
                                         'Prediction')
    taylor_stats18 = sm.taylor_statistics(df["Prediction 7 days past
KT final 300 batch"].squeeze(), df["Original 7 days past final 300
batch"].squeeze(),
                                         'Prediction')
    taylor_stats19 = sm.taylor_statistics(df["Prediction 7 days past
KT final 200 batch"].squeeze(), df["Original 7 days past final 200
```

```
batch"].squeeze(),
                                                'Prediction')
   sdev = np.array([taylor_stats1['sdev'][0],
taylor_stats1['sdev'][1],
                    taylor_stats2['sdev'][1],
taylor_stats3['sdev'][1],
                    taylor_stats4['sdev'][1],
taylor_stats5['sdev'][1],
                    taylor_stats6['sdev'][1],
taylor_stats7['sdev'][1],
                    taylor_stats8['sdev'][1],
taylor_stats9['sdev'][1],
                    taylor_stats10['sdev'][1],
taylor_stats11['sdev'][1],
                    taylor_stats12['sdev'][1],
taylor_stats13['sdev'][1],
                    taylor_stats14['sdev'][1],
taylor_stats15['sdev'][1],
                    taylor_stats16['sdev'][1],
taylor_stats17['sdev'][1],
                    taylor_stats18['sdev'][1],
taylor_stats19['sdev'][1]])
   crmsd = np.array([taylor_stats1['crmsd'][0],
taylor_stats1['crmsd'][1],
                     taylor_stats2['crmsd'][1],
taylor_stats3['crmsd'][1],
                     taylor_stats4['crmsd'][1],
taylor_stats5['crmsd'][1],
                     taylor_stats6['crmsd'][1],
taylor_stats7['crmsd'][1],
                     taylor_stats8['crmsd'][1],
taylor_stats9['crmsd'][1],
                     taylor_stats10['crmsd'][1],
taylor_stats11['crmsd'][1],
                     taylor_stats12['crmsd'][1],
taylor_stats13['crmsd'][1],
                     taylor_stats14['crmsd'][1],
taylor_stats15['crmsd'][1],
                     taylor_stats16['crmsd'][1],
taylor_stats17['crmsd'][1],
                     taylor_stats18['crmsd'][1],
taylor_stats19['crmsd'][1]])
   ccoef = np.array([taylor_stats1['ccoef'][0],
taylor_stats1['ccoef'][1],
                     taylor_stats2['ccoef'][1],
taylor_stats3['ccoef'][1],
                     taylor_stats4['ccoef'][1],
taylor_stats5['ccoef'][1],
                     taylor_stats6['ccoef'][1],
taylor_stats7['ccoef'][1],
                     taylor_stats8['ccoef'][1],
taylor_stats9['ccoef'][1],
                     taylor_stats10['ccoef'][1],
taylor_stats11['ccoef'][1],
                     taylor_stats12['ccoef'][1],
taylor_stats13['ccoef'][1],
                     taylor_stats14['ccoef'][1],
taylor_stats15['ccoef'][1],
                     taylor_stats16['ccoef'][1],
taylor_stats17['ccoef'][1],
                     taylor_stats18['ccoef'][1],
```

```
taylor_stats19['ccoef'][1]])

   label = ['Non-Dimensional Observation', 'M1', 'M2', 'M3', 'M4',
'M5', 'M6', 'M7', 'M8', 'M9', 'M10', 'M11', 'M12',
            'M13', 'M14', 'M15', 'M16', 'M17', 'M18', 'M19']

   sm.taylor_diagram(sdev, crmsd, ccoef, styleOBS = '-',
                     colOBS = 'r', markerobs = 'o',
                     markerLabel=label, titleOBS = 'observation' )

   plt.show()
```

Figura 41: Código para la creación del diagrama de Taylor

4.6 MSE, RMSE, BIAS y Varianza

En el mundo de ML, la precisión lo es todo, cuando un modelo es desarrollado el objetivo es que sea lo más preciso posible, ajustando los parámetros. No importa cuán bueno sea el modelo, existe cierta cantidad de ruido que no se puede eliminar.

Al haber explicado con anterioridad qué es MSE y RMSE, sólo se van a explicar qué son BIAS y varianza.

Error de BIAS es la diferencia entre la predicción esperada del modelo y los valores verdaderos.

- Bajo BIAS: menos suposiciones sobre la forma de la función objetivo.
- Alto BIAS: más suposiciones sobre la forma de la función objetivo.

Error de varianza se refiere a la cantidad que la estimación de la función objetivo cambiará, si se utilizan diferentes datos de entrenamiento.

- Varianza baja: pequeños cambios en la estimación de la función objetivo con cambios en el conjunto de datos de capacitación.
- Varianza alta: grandes cambios en la estimación de la función objetivo con cambios en el conjunto de datos de capacitación.

	RNA sin optimización
7 datos de pasado x4 estaciones	Mean Square Error: 3.822041734687482 Root Mean Square Error: 1.9550042799665381 BIAS: 0.16860037320329546 Variance: 3.6534413614841865

Tabla 4: MSE, RMSE, BIAS y Varianza de RNA sin optimizar

	RNA con optimización (Keras Tuner)
7 datos de pasado x4 estaciones 200 batch	Mean Square Error: 4.019037826733415 Root Mean Square Error: 2.004753807013074 BIAS: -0.022894637068793244 Variance: 4.041932463802208

Tabla 5: MSE, RMSE, BIAS y Varianza de RNA optimizada

Teniendo los valores calculados de cada modelo se puede llegar a ver de una forma más o menos clara, como son los datos obtenidos.

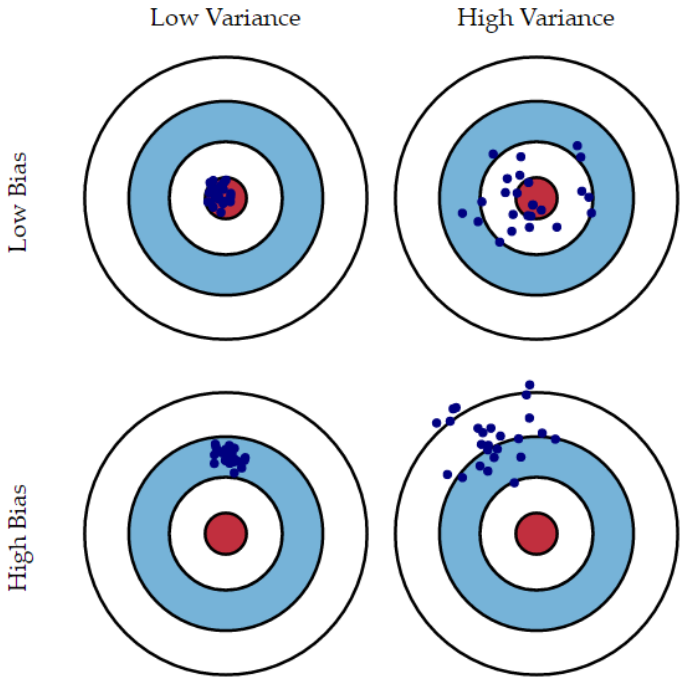

Figura 42: Esquema visual del funcionamiento de BIAS y Varianza

El error BIAS del primer modelo es superior al segundo, ocurriendo lo contrario con la varianza, por lo que el resultado final de estos datos, proporciona una visión muy parecida. Se tienen BIAS bajos con varianzas altas. Según estos datos, no se podría decir qué modelo es mejor, por eso es muy importante complementar con el diagrama de Taylor y las gráficas Q-Q, para tomar una buena decisión.

Como se ha visto en esta sección, se deben realizar diferentes tipos de evaluaciones para poder concretar qué modelo, de todos los creados, es el mejor. Con el diagrama de Taylor, se han determinado que los mejores modelos son M11 y M19. Para decidir cuál de esos dos modelos es mejor, se han evaluado sus gráficos Q-Q, M19 mantiene prácticamente una recta, lo que significa que la distribución del conjunto de datos está cerca de la distribución ideal, mientras que el modelo 11 sub-estima en la cola superior con una mayor diferencia a como lo hace M19, que sobre-estima en la cola inferior. Tras haber realizado la evaluación, se ha determinado, que el mejor modelo es con el diagrama de Taylor, se ha determinado que el mejor modelo es M19.

Para ver las estructuras de los demás modelos de RNA creados, las gráficas de entrenamiento, gráficas Q-Q y valores obtenidos de MSE, RMSE, BIAS y Varianza, ver Anexos: A, B, C, D, E, F, G, H, I y J.

5. Licencia Software y Documental

Llegados a este apartado, se va a proceder a comentar tanto la licencia de software como la licencia documental.

En cuanto a la licencia de software se va a emplear Berkeley Software Distribution (BSD). Se trata de una licencia de software libre permisiva como puede ser OpenSSL o la MIT License. Existen diferentes tipos de licencias, en el caso de este TFG se ha utilizado la licencia “BSD modificada”, “BSD revisada”, “BSD-3” o “BSD de 3 cláusulas” [29].

Figura 43: Logotipo de la licencia BSD

Al igual que sucede en el mundo del software, se tienen que buscar formas de garantizar las libertades asociadas al trabajo elaborado y su inviolabilidad futura. Para garantizar que la libertad esté asociada al documento se buscan *métodos* de protegerla, uno de ellos es la licencia GNU Free Documentation License (GFDL).

Figura 44: Logotipo de la licencia GNU

El propósito de esta Licencia es hacer que en el caso de este TFG sea "gratuito" en el sentido de libertad: para asegurar a todos la libertad efectiva de copiarlo y redistribuirlo, con o sin modificarlo, ya sea comercial o no comercialmente. En segundo lugar, esta licencia preserva para el autor y el editor una forma de obtener crédito por su trabajo, sin ser considerado responsable de las modificaciones realizadas por otros. Es una especie de "copyleft", lo que significa que las obras derivadas del documento deben ser libres en el mismo sentido.

Si por algún motivo se emplea este documento y se modifica, se debe realizar una serie de acciones indicadas en el sitio web oficial de GNU [30]. Tampoco hay que olvidar que este documento, por defecto, está al amparo de la licencia [31], por su inclusión en el Repositorio Institucional de Documentos de la Universidad de Zaragoza: ZAGUAN

6. Conclusiones y Trabajo Futuro

Como conclusiones finales de este TFG destacar que ha sido un desarrollo software, orientado al mundo real, que sin ayuda de los tutores de Nologin Consulting S.L., no hubiera sido posible, por las razones que se van a exponer a continuación. Ha sido una etapa de aprendizaje continuo sobre un campo prácticamente nuevo, ya que en las materias de la universidad se trabaja algo de IA, pero no al mismo nivel que en este TFG.

La IA aplicada a la industria cuenta con la percepción del entorno para encontrar oportunidades. Este TFG se fundamenta en esas bases y este tipo de ideas no podrían surgir si no existiera la colaboración público-privada universidad-empresa.

En el plan de estudios de la Escuela Universitaria Politécnica de Teruel (EUPT) sólo se pueden cursar 12, a lo sumo 18 créditos (Inteligencia Artificial (IA), Almacenes y Minería de Datos (AMD) y Sistema de Ayuda a la Toma de Decisiones (SATD)), relacionados con IA, donde se trabajan aspectos muy básicos de la materia y algunas de sus aplicaciones. Los conocimientos y competencias adquiridas son útiles porque se aprende la terminología, conceptos y arquitectura básicos, pero no al nivel del trabajo desarrollado en este TFG. De ahí el papel tan importante que ha jugado la empresa.

Durante los meses de desarrollo he aprendido a trabajar con herramientas de comunicación como Meet y Rocket.Chat, a documentar con Work On Data y Gitlab y evaluar diferentes soluciones reales para el mismo problema, es decir, he puesto en práctica las competencias transversales adquiridas durante la titulación: a) resolver problemas y tomar decisiones con iniciativa, creatividad y razonamiento crítico y b) aprender de forma continuada y desarrollar estrategias de aprendizaje autónomo. También he avanzado académicamente hablando, ya que he podido profundizar en el amplio y complicado mundo del aprendizaje automático, los almacenes y minería de datos, la estadística, la programación y trabajar, más de cerca, requisitos que se encuentran representados en el modelo de calidad del producto definido por la norma ISO/IEC 25010 (Calidad del producto Software) [32] como son: la eficiencia de desempeño, la compatibilidad, la mantenibilidad y la portabilidad.

Para terminar con las conclusiones, remarcar que cuando no existe una única solución para resolver un problema, se tienen que estudiar todas las posibilidades, analizando su rendimiento, tal y como nos han enseñado en la titulación de Grado en Ingeniería Informática.

Como trabajo futuro sobre este TFG, no estaría de más crear un docker con el software desarrollado para así poder ejecutarlo en diferentes computadoras. Si se quieren introducir nuevos parámetros con los que trabajar, como por ejemplo la precipitación, la racha de viento o su dirección, habría que cambiar la parte del código en la que se especifican los campos a predecir. Además, para mejorar la predicción obtenida en este TFG, sería aconsejable buscar una nueva fuente de datos la cual proporcione más datos desde una fecha inicial y contenga menos datos NaN.

7. Referencias Bibliográficas

[1] GAMEZ, MARIA, 2021, Objetivos y metas de desarrollo sostenible. *Desarrollo Sostenible* [online]. 2021. [Accessed 14 June 2021]. Available from: https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/

[2] AEMET OpenData, 2021. *Opendata.aemet.es* [online]

[3] RAMÍREZ, Misael Dario Rosales, et al. Redes neuronales en la predicción de micro-clima, zona de estudio La Hechicera Mérida, Venezuela. *Publicaciones en ciencias y tecnología*, 2017, vol. 11, no 2, p. 47-61.

[4] JOHNSTONE, Charles; SULUNGU, Emmanuel D. Application of neural network in prediction of temperature: a review. *Neural Computing and Applications*, 2021, p. 1-12.

[5] PARK, Inyoung, et al. Temperature prediction using the missing data refinement model based on a long short-term memory neural network. *Atmosphere*, 2019, vol. 10, no 11, p. 718.

[6] IRIGOYEN MANCHO, Miguel. Reconocimiento de patrones meteorológicos mediante técnicas neuronales y estadísticas. 2011.

[7] RASP, Stephan; LERCH, Sebastian. Neural networks for postprocessing ensemble weather forecasts. *Monthly Weather Review*, 2018, vol. 146, no 11, p. 3885-3900.

[8] IZAURIETA, Fernando; SAAVEDRA, Carlos. Redes neuronales artificiales. *Departamento de Física, Universidad de Concepción Chile*, 2000.

[9] MATICH, Damián Jorge. Redes Neuronales: Conceptos básicos y aplicaciones. *Universidad Tecnológica Nacional, México*, 2001, vol. 41.

[10] Brownlee, J. (2021). How to Choose an Activation Function for Deep Learning. Retrieved 11 May 2021, from https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/

[11] Profesor e investigador en Inteligencia Artificial y Supercomputación. (2021). Retrieved from https://torres.ai/

[12] Keras LSTM tutorial – How to easily build a powerful deep learning language model – Adventures in Machine Learning. (2021). Retrieved 2021, from https://adventuresinmachinelearning.com/keras-lstm-tutorial/

[13] LSTM Recurrent Neural Network Keras Example. (2021). Retrieved 2021, from https://towardsdatascience.com/machine-learning-recurrent-neural-networks-and-long-short-term-memory-lstm-python-keras-example-86001ceaaebc

[14] Understanding LSTM Networks -- colah's blog. (2021). Retrieved 2021, from https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[15] Illustrated Guide to LSTM's and GRU's: A step by step explanation. (2021). Retrieved 2021, from https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

[16] A Quick Deep Learning Recipe: Time Series Forecasting with Keras in Python. (2021). Retrieved 2021, from https://towardsdatascience.com/a-quick-deep-learning-recipe-time-series-forecasting-with-keras-in-python-f759923ba64

[17] Team, K. (2021). Keras: the Python deep learning API. Retrieved 2021, from https://keras.io/

[18] TensorFlow. (2021). Retrieved 2021, from https://www.tensorflow.org/

[19] Swagger UI. (2021). Retrieved 2021, from https://opendata.aemet.es/dist/index.html

[20] AEMET OpenData. (2021). Retrieved 2021, from https://opendata.aemet.es/centrodedescargas/altaUsuario

[21] NoSQL - Wikipedia, la enciclopedia libre, 2021. *Es.wikipedia.org* [online]

[22] Explicación sobre las bases de datos NoSQL, 2021. *MongoDB* [online]

[23] Pronóstico de Series Temporales con Redes Neuronales en Python | Aprende Machine Learning. (2021). Retrieved 2021, from https://www.aprendemachinelearning.com/pronostico-de-series-temporales-con-redes-neuronales-en-python/

[24] OVANDO, Gustavo; BOCCO, Mónica; SAYAGO, Silvina. Redes neuronales para modelar predicción de heladas. *Agricultura Técnica*, 2005, vol. 65, no 1, p. 65-73.

[25] LABAJO, Angel L.; LABAJO, José L. UNA RED NEURONAL COMO HERRAMIENTA DE PREDICCIÓN DE VARIABLES CLIMÁTICAS: APLICACIÓN A LA TEMPERATURA MÍNIMA MEDIA MENSUAL EN CASTILLA y LEÓN. *Acta de las Jornadas Científicas de la Asociación Meteorológica Española*, 2020, no 31

[26] Brownlee, J. (2021). Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras. Retrieved 2021, from https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/

[27] Time Series Analysis with LSTM using Python's Keras Library. (2021). Retrieved 2021, from https://stackabuse.com/time-series-analysis-with-lstm-using-pythons-keras-library/

[28] PeterRochford/SkillMetrics, 2021. *GitHub* [online]

[29] The GNU Operating System and the Free Software Movement, 2021. Gnu.org [online]

[30] Licencia BSD - Wikipedia, la enciclopedia libre, 2021. *Es.wikipedia.org* [online]

[31] Creative Commons — Reconocimiento-NoComercial-SinObraDerivada 3.0 España — CC BY-NC-ND 3.0 ES, 2021. *Creativecommons.org* [online],

[32] DISCERN, CGM, 25000, LOS, PA, EL, CHAPP, SICAMAN and PROMETEUS DELFOS 1.0.0, NUEVO CERTIFICADO ISO/IEC 25000, 2021, ISO 25010. *Iso25000.com* [online]. 2021. [Accessed 14 June 2021]. Available from: https://iso25000.com/index.php/normas-iso-25000/iso-25010#:~:text=ISO%2FIEC%2025010&text=La%20calidad%20del%20producto%20software, seguridad%2C%20mantenibilidad%2C%20etc.)

Anexos

En este anexo se presentan las diferentes gráficas de los modelos de RNA creados. Han sido un total de 19 modelos de los cuales 2, ya han sido comentados por ser los mejores modelos obtenidos. Ahora se van a presentar los modelos restantes.

Anexo A: Estructura de las RNA sin optimizar

A continuación, se muestran las estructuras de los modelos de RNA creados sin hacer uso de la optimización de Keras Tuner.

RNA con 2+1 días y 1 estación meteorológica

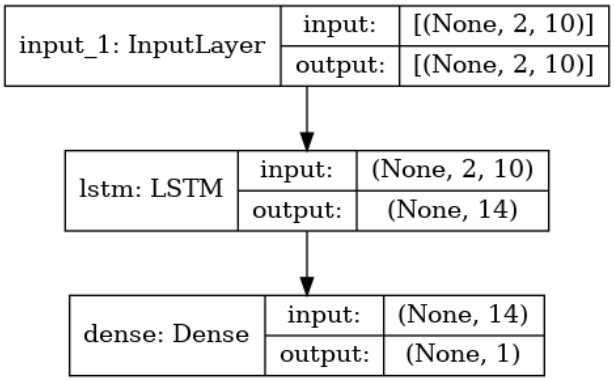

Figura 45: Estructura RNA con 2+1 días y 1 estación meteorológica sin optimizar

RNA con 5+1 días y 1 estación meteorológica

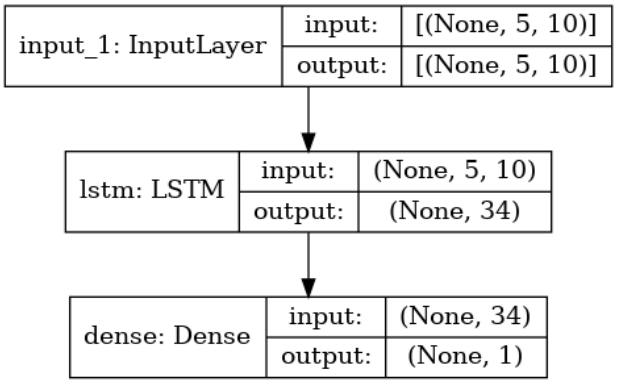

Figura 46: Estructura RNA con 5+1 días y 1 estación meteorológica sin optimizar

RNA con 7+1 días y 1 estación meteorológica

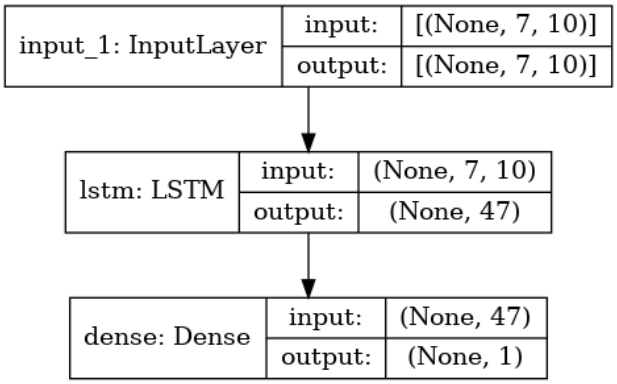

Figura 47: Estructura RNA con 7+1 días y 1 estación meteorológica sin optimizar

RNA con 10+1 días y 1 estación meteorológica

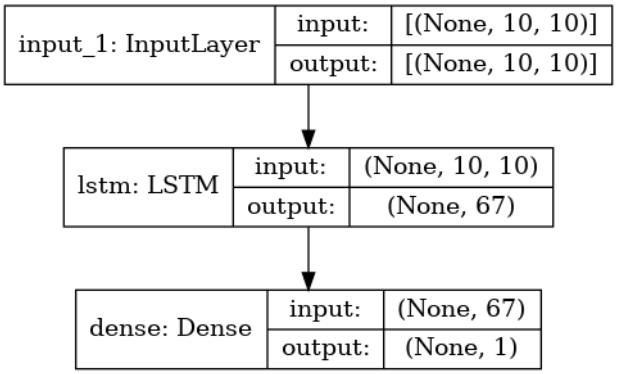

Figura 48: Estructura RNA con 10+1 días y 1 estación meteorológica sin optimizar

RNA con 2+1 días y 4 estaciones meteorológicas

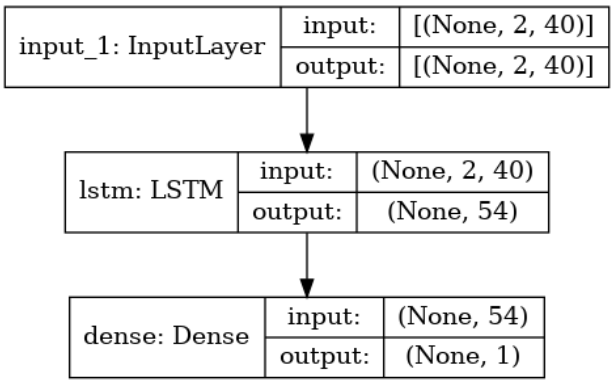

Figura 49: Estructura RNA con 2+1 días y 4 estaciones meteorológicas sin optimizar

RNA con 5+1 días y 4 estaciones meteorológicas

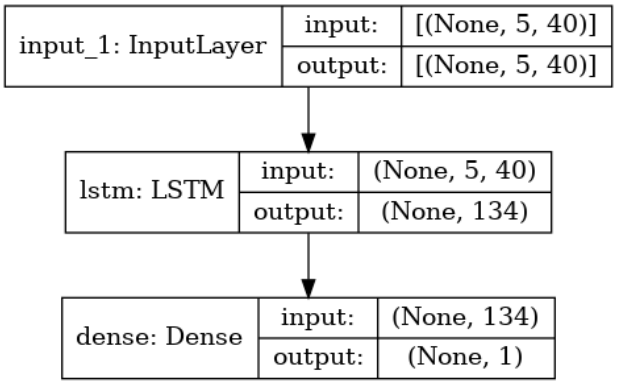

Figura 50: Estructura RNA con 5+1 días y 4 estaciones meteorológicas sin optimizar

RNA con 10+1 días y 4 estaciones meteorológicas

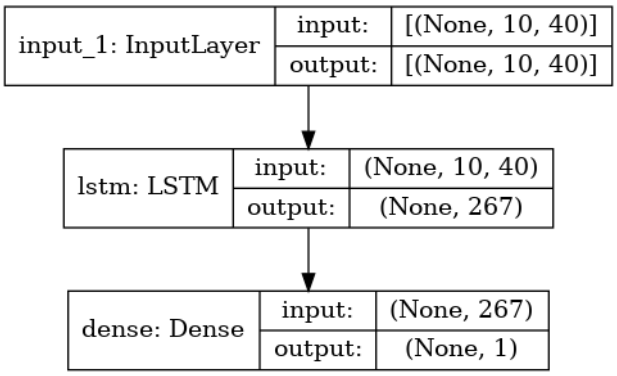

Figura 51: Estructura RNA con 10+1 días y 4 estaciones meteorológicas sin optimizar

RNA con 7+1 días, 4 estaciones meteorológicas y 300 de tamaño de lote

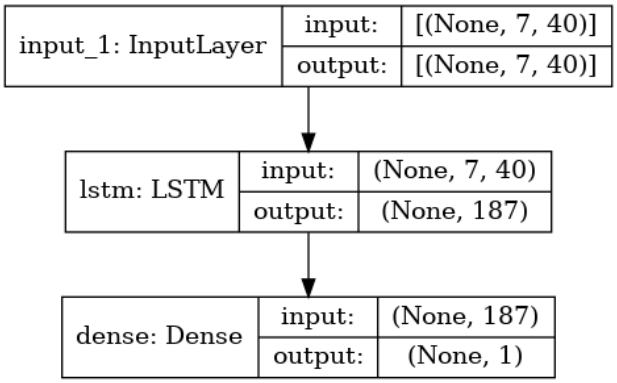

Figura 52: Estructura RNA con 7+1 días, 4 estaciones meteorológicas y 300 de tamaño de lote

Anexo B: Entrenamiento de las RNA sin optimizar

En este punto se van a mostrar los entrenamientos de cada una de las RNA sin optimizar y con las diferentes configuraciones que se han llevado a cabo.

RNA con 2+1 días y 1 estación meteorológica

```
Epoch 9/20
1476/1476 [==============================] - 4s 3ms/step - loss: 0.0027 - val_loss: 0.0026
Epoch 10/20
1476/1476 [==============================] - 4s 3ms/step - loss: 0.0027 - val_loss: 0.0025
Epoch 11/20
1476/1476 [==============================] - 4s 3ms/step - loss: 0.0027 - val_loss: 0.0025
Epoch 12/20
1476/1476 [==============================] - 4s 3ms/step - loss: 0.0026 - val_loss: 0.0025
Epoch 13/20
1476/1476 [==============================] - 4s 3ms/step - loss: 0.0026 - val_loss: 0.0025
Epoch 14/20
1476/1476 [==============================] - 4s 3ms/step - loss: 0.0026 - val_loss: 0.0025
Epoch 15/20
1476/1476 [==============================] - 4s 3ms/step - loss: 0.0026 - val_loss: 0.0024
Epoch 16/20
1476/1476 [==============================] - 4s 3ms/step - loss: 0.0025 - val_loss: 0.0024
Epoch 17/20
1476/1476 [==============================] - 4s 3ms/step - loss: 0.0025 - val_loss: 0.0024
Epoch 18/20
1476/1476 [==============================] - 4s 3ms/step - loss: 0.0025 - val_loss: 0.0024
Epoch 19/20
1476/1476 [==============================] - 4s 3ms/step - loss: 0.0025 - val_loss: 0.0024
Epoch 20/20
1476/1476 [==============================] - 4s 3ms/step - loss: 0.0025 - val_loss: 0.0024
```

Figura 53: Entrenamiento RNA con 2+1 días y 1 estación meteorológica sin optimizar

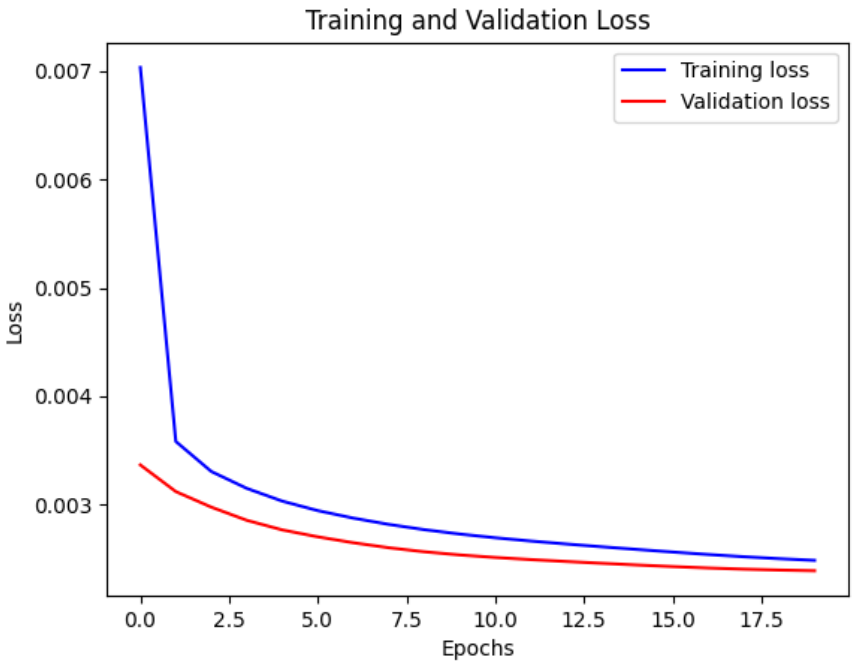

Figura 54: Gráfica de pérdida RNA con 2+1 días y 1 estación meteorológica sin optimizar

RNA con 5+1 días y 1 estación meteorológica

```
Epoch 9/20
1440/1440 [==============================] - 7s 5ms/step - loss: 0.0026 - val_loss: 0.0026
Epoch 10/20
1440/1440 [==============================] - 5s 4ms/step - loss: 0.0026 - val_loss: 0.0025
Epoch 11/20
1440/1440 [==============================] - 5s 4ms/step - loss: 0.0025 - val_loss: 0.0025
Epoch 12/20
1440/1440 [==============================] - 7s 5ms/step - loss: 0.0025 - val_loss: 0.0025
Epoch 13/20
1440/1440 [==============================] - 6s 4ms/step - loss: 0.0025 - val_loss: 0.0025
Epoch 14/20
1440/1440 [==============================] - 5s 4ms/step - loss: 0.0024 - val_loss: 0.0025
Epoch 15/20
1440/1440 [==============================] - 5s 4ms/step - loss: 0.0024 - val_loss: 0.0025
Epoch 16/20
1440/1440 [==============================] - 5s 4ms/step - loss: 0.0024 - val_loss: 0.0025
Epoch 17/20
1440/1440 [==============================] - 5s 4ms/step - loss: 0.0024 - val_loss: 0.0025
Epoch 18/20
1440/1440 [==============================] - 5s 4ms/step - loss: 0.0024 - val_loss: 0.0025
Epoch 19/20
1440/1440 [==============================] - 5s 4ms/step - loss: 0.0024 - val_loss: 0.0025
Epoch 20/20
1440/1440 [==============================] - 5s 4ms/step - loss: 0.0024 - val_loss: 0.0025
```

Figura 55: Entrenamiento RNA con 5+1 días y 1 estación meteorológica sin optimizar

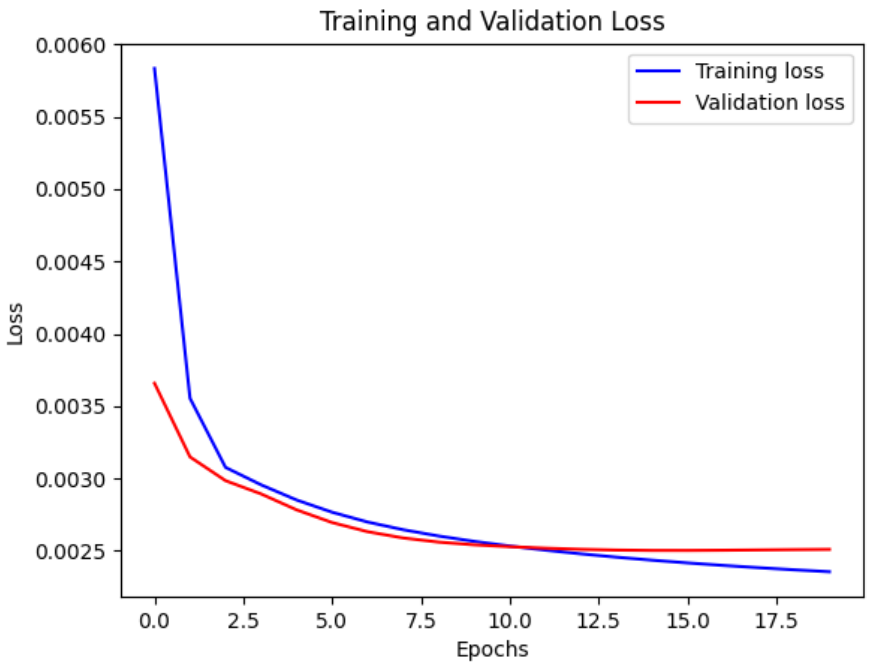

Figura 56: Gráfica de pérdida RNA con 5+1 días y 1 estación meteorológica sin optimizar

RNA con 7+1 días y 1 estación meteorológica

```
Epoch 9/20
1418/1418 [==============================] - 6s 4ms/step - loss: 0.0026 - val_loss: 0.0026
Epoch 10/20
1418/1418 [==============================] - 6s 4ms/step - loss: 0.0025 - val_loss: 0.0026
Epoch 11/20
1418/1418 [==============================] - 6s 4ms/step - loss: 0.0025 - val_loss: 0.0025
Epoch 12/20
1418/1418 [==============================] - 6s 4ms/step - loss: 0.0024 - val_loss: 0.0025
Epoch 13/20
1418/1418 [==============================] - 6s 4ms/step - loss: 0.0024 - val_loss: 0.0025
Epoch 14/20
1418/1418 [==============================] - 6s 4ms/step - loss: 0.0024 - val_loss: 0.0025
Epoch 15/20
1418/1418 [==============================] - 6s 4ms/step - loss: 0.0024 - val_loss: 0.0025
Epoch 16/20
1418/1418 [==============================] - 6s 4ms/step - loss: 0.0023 - val_loss: 0.0025
Epoch 17/20
1418/1418 [==============================] - 6s 4ms/step - loss: 0.0023 - val_loss: 0.0025
Epoch 18/20
1418/1418 [==============================] - 6s 4ms/step - loss: 0.0023 - val_loss: 0.0025
Epoch 19/20
1418/1418 [==============================] - 6s 4ms/step - loss: 0.0023 - val_loss: 0.0026
Epoch 20/20
1418/1418 [==============================] - 6s 4ms/step - loss: 0.0023 - val_loss: 0.0026
```

Figura 57: Entrenamiento RNA con 7+1 días y 1 estación meteorológica sin optimizar

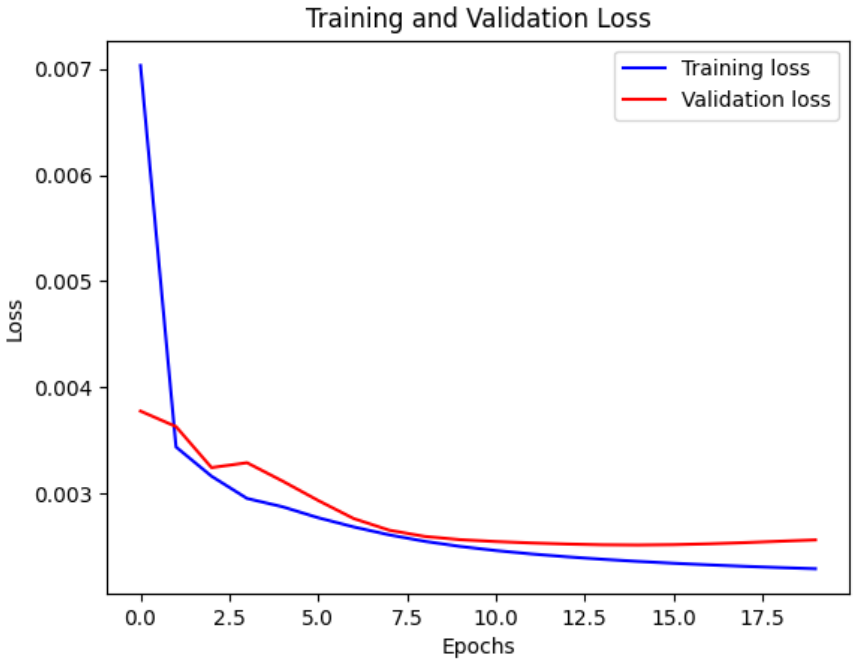

Figura 58: Gráfica de pérdida RNA con 7+1 días y 1 estación meteorológica sin optimizar

RNA con 10+1 días y 1 estación meteorológica

```
Epoch 9/20
1386/1386 [==============================] - 8s 5ms/step - loss: 0.0025 - val_loss: 0.0026
Epoch 10/20
1386/1386 [==============================] - 8s 5ms/step - loss: 0.0025 - val_loss: 0.0026
Epoch 11/20
1386/1386 [==============================] - 8s 5ms/step - loss: 0.0024 - val_loss: 0.0026
Epoch 12/20
1386/1386 [==============================] - 8s 5ms/step - loss: 0.0024 - val_loss: 0.0026
Epoch 13/20
1386/1386 [==============================] - 8s 5ms/step - loss: 0.0024 - val_loss: 0.0027
Epoch 14/20
1386/1386 [==============================] - 8s 5ms/step - loss: 0.0024 - val_loss: 0.0027
Epoch 15/20
1386/1386 [==============================] - 8s 5ms/step - loss: 0.0024 - val_loss: 0.0028
Epoch 16/20
1386/1386 [==============================] - 8s 6ms/step - loss: 0.0024 - val_loss: 0.0028
Epoch 17/20
1386/1386 [==============================] - 8s 5ms/step - loss: 0.0024 - val_loss: 0.0027
Epoch 18/20
1386/1386 [==============================] - 8s 5ms/step - loss: 0.0024 - val_loss: 0.0027
Epoch 19/20
1386/1386 [==============================] - 8s 5ms/step - loss: 0.0024 - val_loss: 0.0027
Epoch 20/20
1386/1386 [==============================] - 8s 5ms/step - loss: 0.0023 - val_loss: 0.0026
```

Figura 59: Entrenamiento RNA con 10+1 días y 1 estación meteorológica sin optimizar

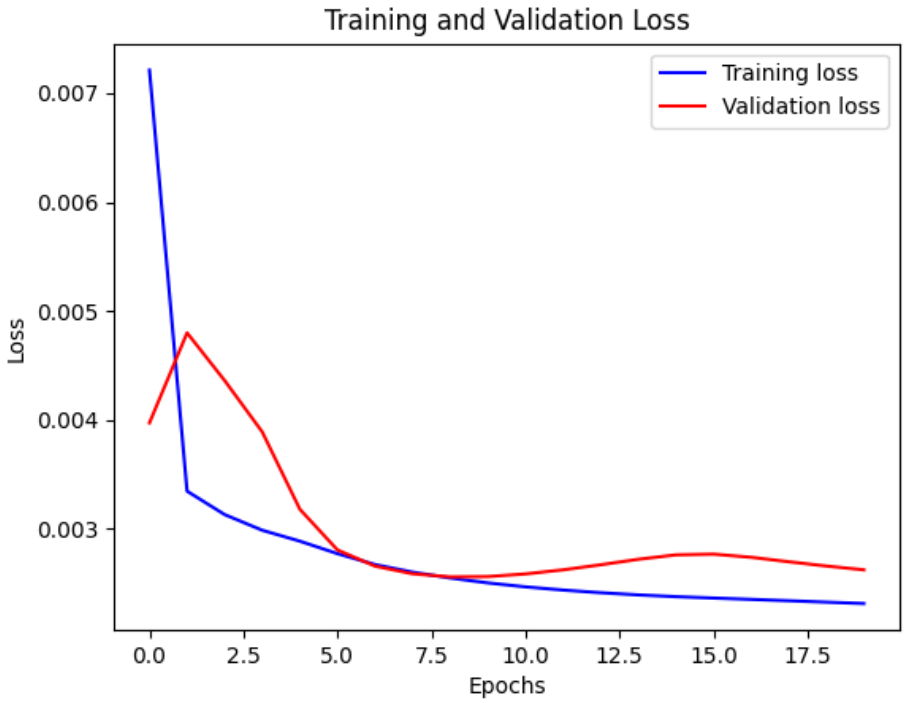

Figura 60: Gráfica de pérdida RNA con 10+1 días y 1 estación meteorológica sin optimizar

RNA con 2+1 días y 4 estaciones meteorológicas

```
Epoch 9/20
716/716 [==============================] - 2s 3ms/step - loss: 0.0032 - val_loss: 0.0029
Epoch 10/20
716/716 [==============================] - 2s 3ms/step - loss: 0.0031 - val_loss: 0.0029
Epoch 11/20
716/716 [==============================] - 3s 4ms/step - loss: 0.0031 - val_loss: 0.0029
Epoch 12/20
716/716 [==============================] - 2s 3ms/step - loss: 0.0030 - val_loss: 0.0029
Epoch 13/20
716/716 [==============================] - 2s 3ms/step - loss: 0.0030 - val_loss: 0.0028
Epoch 14/20
716/716 [==============================] - 2s 3ms/step - loss: 0.0029 - val_loss: 0.0028
Epoch 15/20
716/716 [==============================] - 2s 3ms/step - loss: 0.0029 - val_loss: 0.0028
Epoch 16/20
716/716 [==============================] - 2s 3ms/step - loss: 0.0028 - val_loss: 0.0028
Epoch 17/20
716/716 [==============================] - 2s 3ms/step - loss: 0.0028 - val_loss: 0.0028
Epoch 18/20
716/716 [==============================] - 2s 3ms/step - loss: 0.0028 - val_loss: 0.0027
Epoch 19/20
716/716 [==============================] - 3s 4ms/step - loss: 0.0027 - val_loss: 0.0027
Epoch 20/20
716/716 [==============================] - 2s 3ms/step - loss: 0.0027 - val_loss: 0.0027
```

Figura 61: Entrenamiento RNA con 2+1 días y 4 estaciones meteorológicas sin optimizar

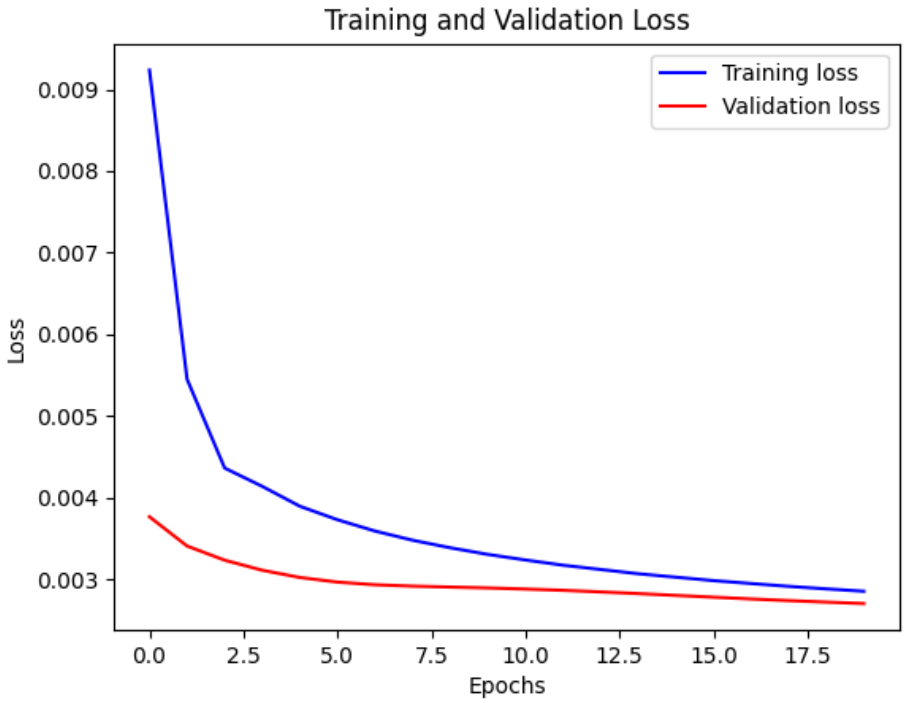

Figura 62: Gráfica de pérdida RNA con 2+1 días y 4 estaciones meteorológicas sin optimizar

RNA con 5+1 días y 4 estaciones meteorológicas

```
Epoch 9/20
516/516 [==============================] - 3s 6ms/step - loss: 0.0030 - val_loss: 0.0028
Epoch 10/20
516/516 [==============================] - 3s 6ms/step - loss: 0.0029 - val_loss: 0.0027
Epoch 11/20
516/516 [==============================] - 3s 6ms/step - loss: 0.0029 - val_loss: 0.0027
Epoch 12/20
516/516 [==============================] - 3s 6ms/step - loss: 0.0028 - val_loss: 0.0027
Epoch 13/20
516/516 [==============================] - 3s 6ms/step - loss: 0.0027 - val_loss: 0.0027
Epoch 14/20
516/516 [==============================] - 3s 6ms/step - loss: 0.0027 - val_loss: 0.0027
Epoch 15/20
516/516 [==============================] - 3s 6ms/step - loss: 0.0026 - val_loss: 0.0027
Epoch 16/20
516/516 [==============================] - 3s 6ms/step - loss: 0.0026 - val_loss: 0.0027
Epoch 17/20
516/516 [==============================] - 3s 6ms/step - loss: 0.0026 - val_loss: 0.0027
Epoch 18/20
516/516 [==============================] - 3s 6ms/step - loss: 0.0025 - val_loss: 0.0027
Epoch 19/20
516/516 [==============================] - 3s 6ms/step - loss: 0.0025 - val_loss: 0.0027
Epoch 20/20
516/516 [==============================] - 3s 6ms/step - loss: 0.0025 - val_loss: 0.0027
```

Figura 63: Entrenamiento RNA con 5+1 días y 4 estaciones meteorológicas sin optimizar

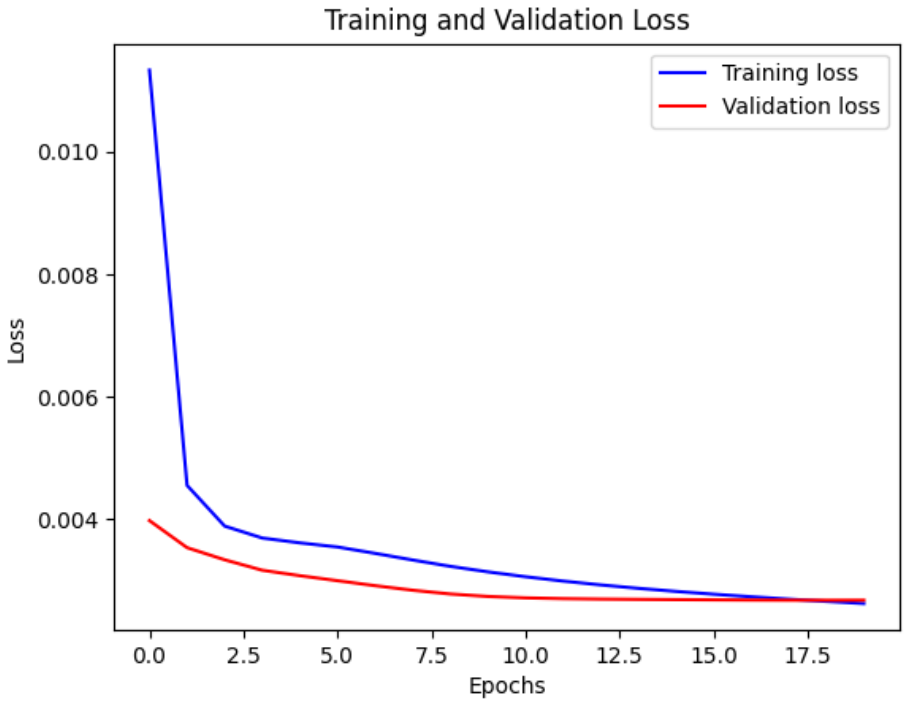

Figura 64: Gráfica de pérdida RNA con 5+1 días y 4 estaciones meteorológicas sin optimizar

RNA con 10+1 días y 4 estaciones meteorológicas

```
Epoch 9/20
353/353 [==============================] - 7s 20ms/step - loss: 0.0033 - val_loss: 0.0038
Epoch 10/20
353/353 [==============================] - 7s 19ms/step - loss: 0.0032 - val_loss: 0.0038
Epoch 11/20
353/353 [==============================] - 7s 19ms/step - loss: 0.0032 - val_loss: 0.0038
Epoch 12/20
353/353 [==============================] - 8s 22ms/step - loss: 0.0031 - val_loss: 0.0037
Epoch 13/20
353/353 [==============================] - 7s 20ms/step - loss: 0.0031 - val_loss: 0.0037
Epoch 14/20
353/353 [==============================] - 7s 20ms/step - loss: 0.0030 - val_loss: 0.0037
Epoch 15/20
353/353 [==============================] - 7s 20ms/step - loss: 0.0030 - val_loss: 0.0037
Epoch 16/20
353/353 [==============================] - 7s 20ms/step - loss: 0.0029 - val_loss: 0.0036
Epoch 17/20
353/353 [==============================] - 7s 20ms/step - loss: 0.0028 - val_loss: 0.0036
Epoch 18/20
353/353 [==============================] - 7s 20ms/step - loss: 0.0028 - val_loss: 0.0035
Epoch 19/20
353/353 [==============================] - 7s 20ms/step - loss: 0.0027 - val_loss: 0.0035
Epoch 20/20
353/353 [==============================] - 7s 20ms/step - loss: 0.0027 - val_loss: 0.0034
```

Figura 65: Entrenamiento RNA con 10+1 días y 4 estaciones meteorológicas sin optimizar

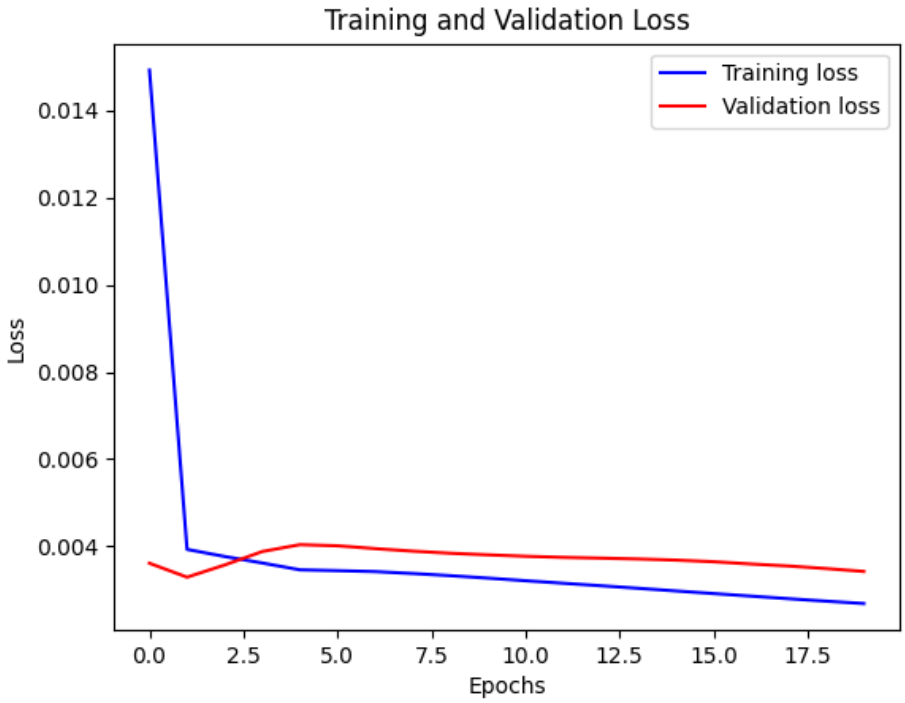

Figura 66: Gráfica de pérdida RNA con 10+1 días y 4 estaciones meteorológicas sin optimizar

RNA con 7+1 días, 4 estaciones meteorológicas y 300 de tamaño de lote

```
Epoch 9/20
15/15 [==============================] - 1s 54ms/step - loss: 0.0047 - val_loss: 0.0057
Epoch 10/20
15/15 [==============================] - 1s 54ms/step - loss: 0.0045 - val_loss: 0.0055
Epoch 11/20
15/15 [==============================] - 1s 54ms/step - loss: 0.0044 - val_loss: 0.0053
Epoch 12/20
15/15 [==============================] - 1s 55ms/step - loss: 0.0042 - val_loss: 0.0051
Epoch 13/20
15/15 [==============================] - 1s 55ms/step - loss: 0.0041 - val_loss: 0.0049
Epoch 14/20
15/15 [==============================] - 1s 56ms/step - loss: 0.0039 - val_loss: 0.0047
Epoch 15/20
15/15 [==============================] - 1s 55ms/step - loss: 0.0038 - val_loss: 0.0045
Epoch 16/20
15/15 [==============================] - 1s 54ms/step - loss: 0.0037 - val_loss: 0.0043
Epoch 17/20
15/15 [==============================] - 1s 55ms/step - loss: 0.0036 - val_loss: 0.0042
Epoch 18/20
15/15 [==============================] - 1s 55ms/step - loss: 0.0035 - val_loss: 0.0041
Epoch 19/20
15/15 [==============================] - 1s 55ms/step - loss: 0.0034 - val_loss: 0.0039
Epoch 20/20
15/15 [==============================] - 1s 54ms/step - loss: 0.0033 - val_loss: 0.0039
```

Figura 67: Entrenamiento RNA con 7+1 días, 4 estaciones meteorológicas y 300 de tamaño de lote sin optimizar

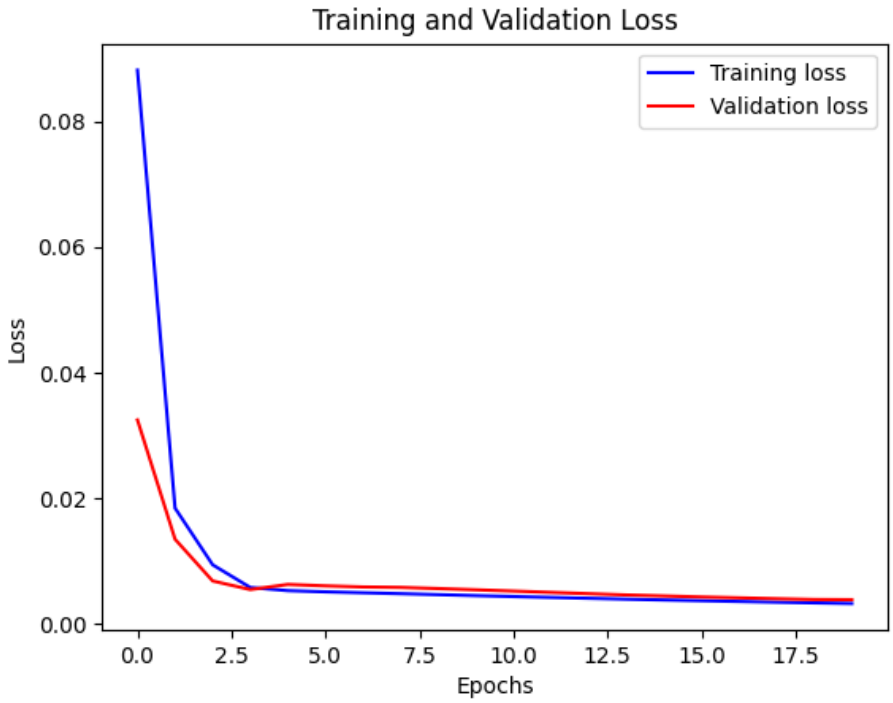

Figura 68: Gráfica de pérdida RNA con 7+1 días, 4 estaciones meteorológicas y 300 de tamaño de lote sin optimizar

Anexo C: Comparación datos reales y datos predichos de RNA sin optimizar

En esta sección se van a mostrar las diferentes gráficas donde se comparan los datos reales de las temperaturas medias con las temperaturas predichas por cada RNA creada.

RNA con 2+1 días y 1 estación meteorológica

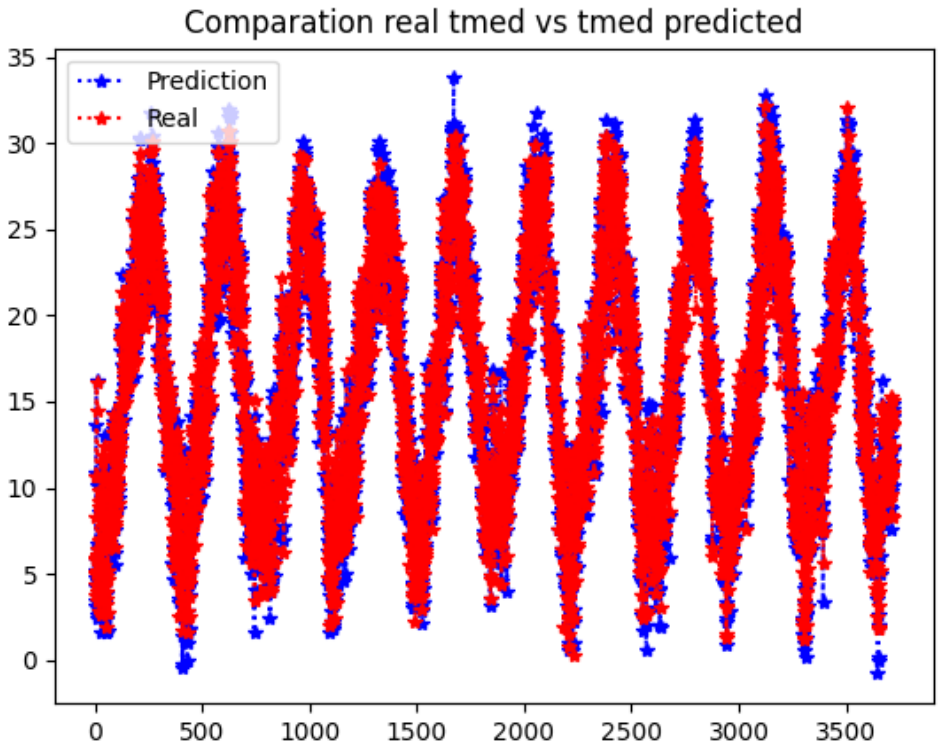

Figura 69: Gráfica de comparación de datos RNA con 2+1 días y 1 estación meteorológica sin optimizar

	Original	Prediction
0	3.2	3.583700
1	5.8	5.394649
2	4.8	5.803415
3	4.3	5.031893
4	3.5	5.047049
...	...	...
3717	12.3	13.988044
3718	14.6	14.305723
3719	14.9	14.485455
3720	14.2	14.942598
3721	14.2	13.621502

[3722 rows x 2 columns]

Figura 70: Comparación de datos RNA con 2+1 días y 1 estación meteorológica sin optimizar

RNA con 5+1 días y 1 estación meteorológica

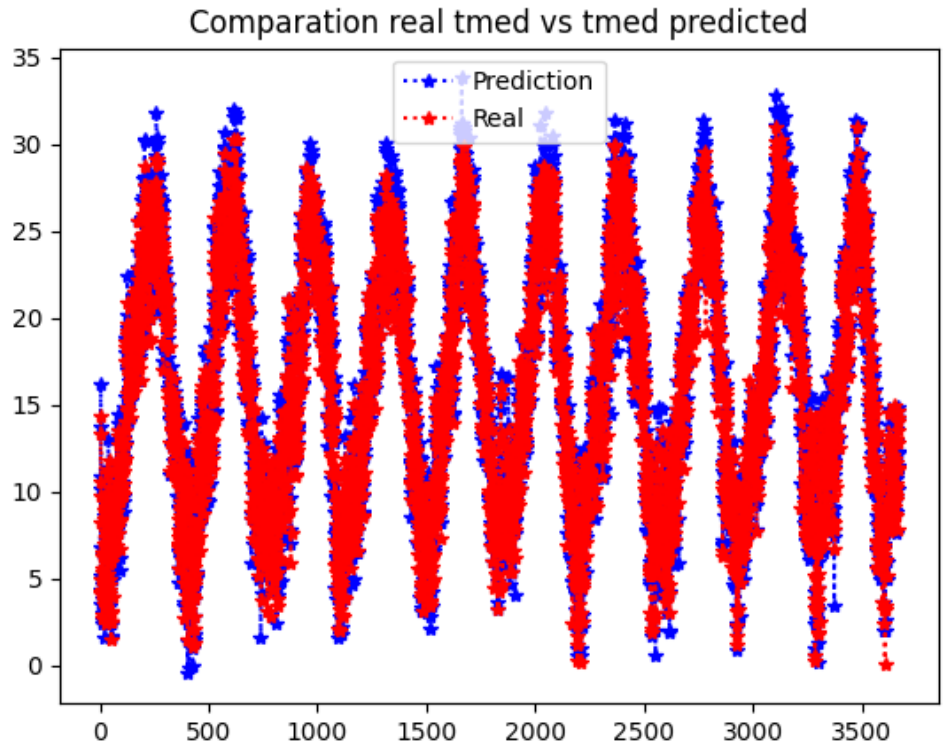

Figura 71: Gráfica de comparación de datos RNA con 5+1 días y 1 estación meteorológica sin optimizar

	Original 5 days past	Prediction 5 days past
0	4.3	4.964064
1	3.5	4.154237
2	5.2	4.220656
3	10.9	8.218985
4	13.7	10.606447
...	...	...
3666	12.3	12.835330
3667	14.6	14.041592
3668	14.9	14.262041
3669	14.2	14.658355
3670	14.2	13.488866

Figura 72: Comparación de datos RNA con 5+1 días y 1 estación meteorológica sin optimizar

RNA con 7+1 días y 1 estación meteorológica

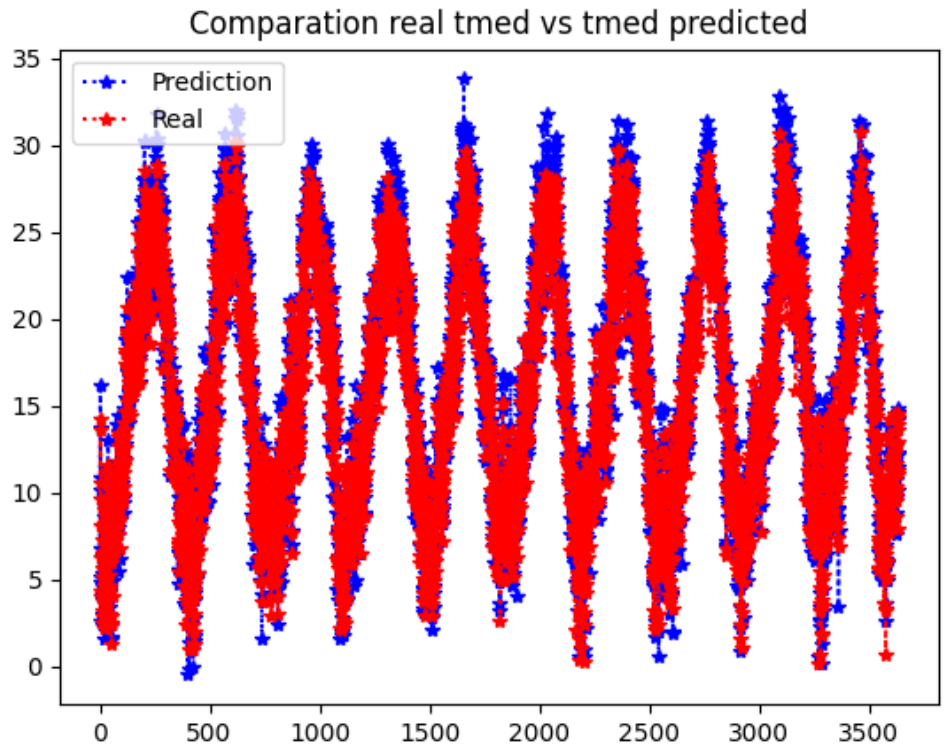

Figura 73: Gráfica de comparación de datos RNA con 7+1 días y 1 estación meteorológica sin optimizar

	Original 7 days past	Prediction 7 days past
0	5.2	4.324853
1	10.9	8.079282
2	13.7	10.591175
3	16.2	13.622806
4	10.2	14.191107
...	...	...
3632	12.3	12.904532
3633	14.6	13.925821
3634	14.9	14.354973
3635	14.2	14.522936
3636	14.2	13.658095

[3637 rows x 2 columns]

Figura 74: Comparación de datos RNA con 7+1 días y 1 estación meteorológica sin optimizar

RNA con 10+1 días y 1 estación meteorológica

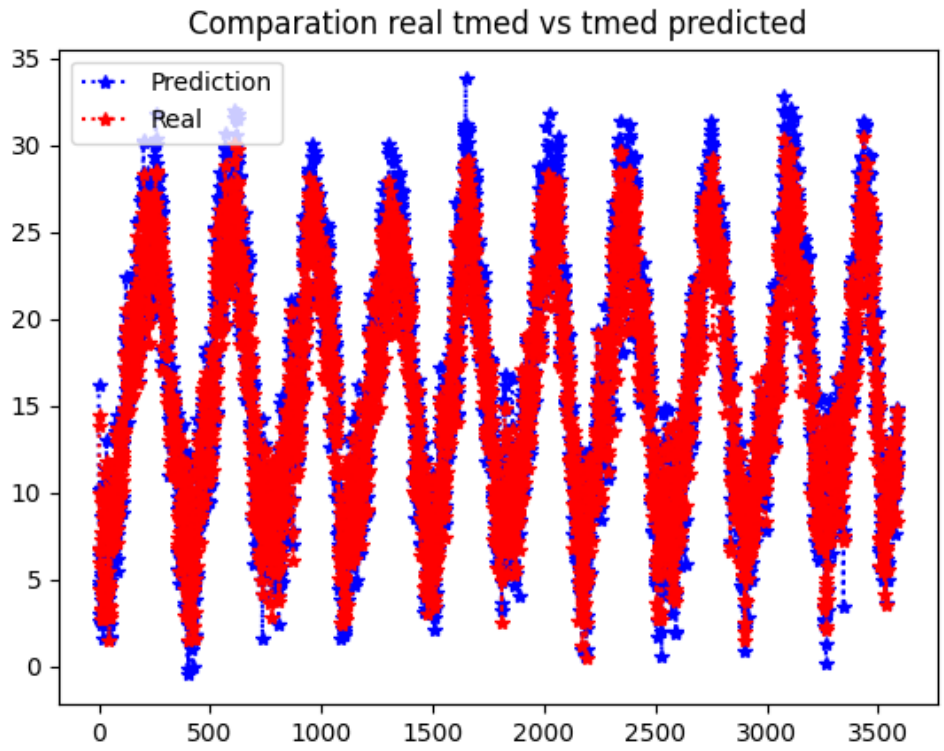

Figura 75: Gráfica de comparación de datos RNA con 10+1 días y 1 estación meteorológica sin optimizar

	Original	Prediction
0	16.2	13.829622
1	10.2	14.436267
2	6.8	9.955258
3	2.6	6.838455
4	4.8	4.450349
...	...	...
3584	12.3	13.043367
3585	14.6	14.172271
3586	14.9	14.600101
3587	14.2	14.760103
3588	14.2	13.680773

[3589 rows x 2 columns]

Figura 76: Comparación de datos RNA con 10+1 días y 1 estación meteorológica sin optimizar

RNA con 2+1 días y 4 estaciones meteorológicas

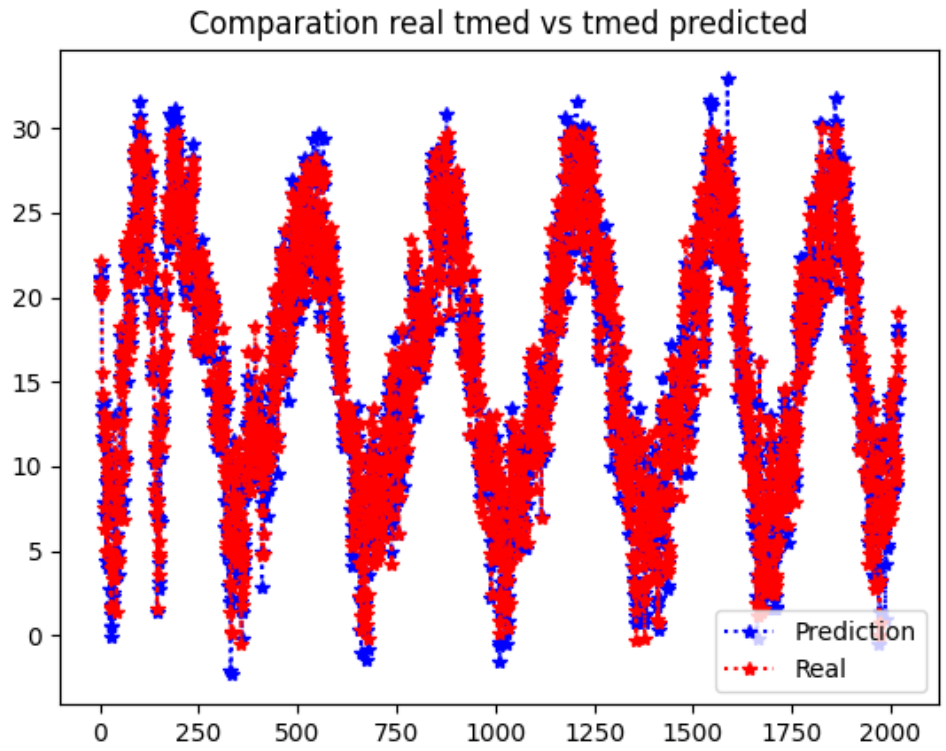

Figura 77: Gráfica de comparación de datos RNA con 2+1 días y 4 estaciones meteorológicas sin optimizar

	Original 2 days past final	Prediction 2 days past final
0	21.1	20.903467
1	21.2	20.621169
2	20.4	22.138412
3	21.1	20.381334
4	20.3	20.114534
...	...	...
2016	14.0	14.838764
2017	15.0	15.841804
2018	17.8	16.523535
2019	18.3	17.416865
2020	17.9	19.146088

[2021 rows x 2 columns]

Figura 78: Comparación de datos RNA con 2+1 días y 4 estaciones meteorológicas sin optimizar

RNA con 5+1 días y 4 estaciones meteorológicas

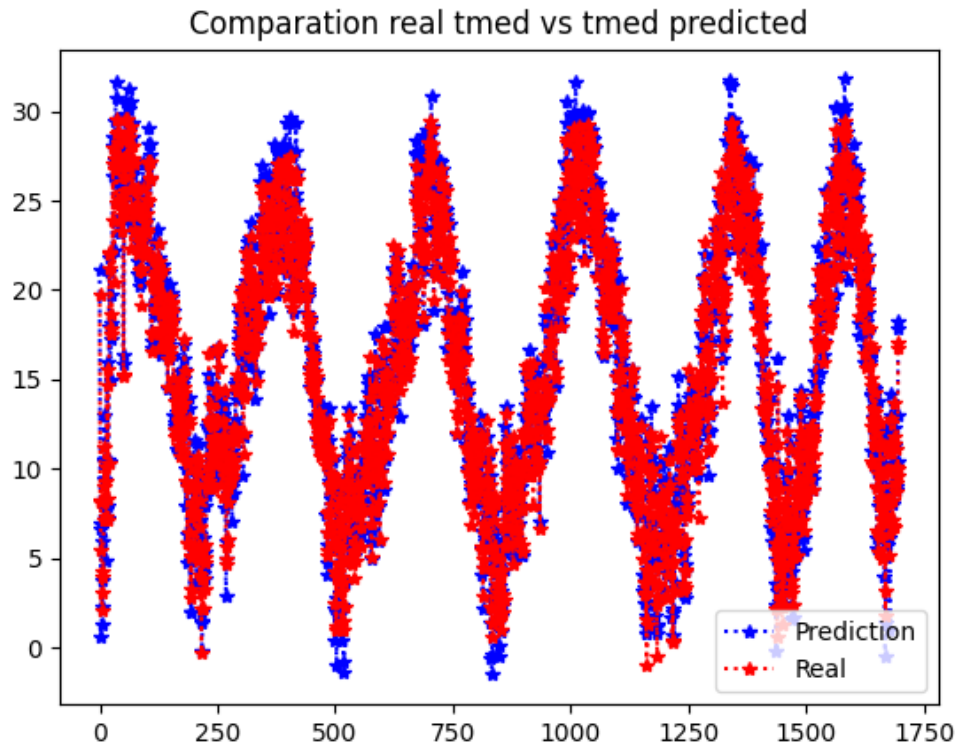

Figura 79: Gráfica de comparación de datos RNA con 5+1 días y 4 estaciones meteorológicas sin optimizar

	Original 5 days past final	Prediction 5 days past final
0	21.1	19.793099
1	7.0	8.279406
2	6.7	8.157811
3	0.6	5.466463
4	1.3	2.066866
...	...	...
1693	9.0	8.768196
1694	10.4	10.031201
1695	13.0	9.663652
1696	18.3	16.764462
1697	17.9	17.051710

[1698 rows x 2 columns]

Figura 80: Comparación de datos RNA con 5+1 días y 4 estaciones meteorológicas sin optimizar

RNA con 10+1 días y 4 estaciones meteorológicas

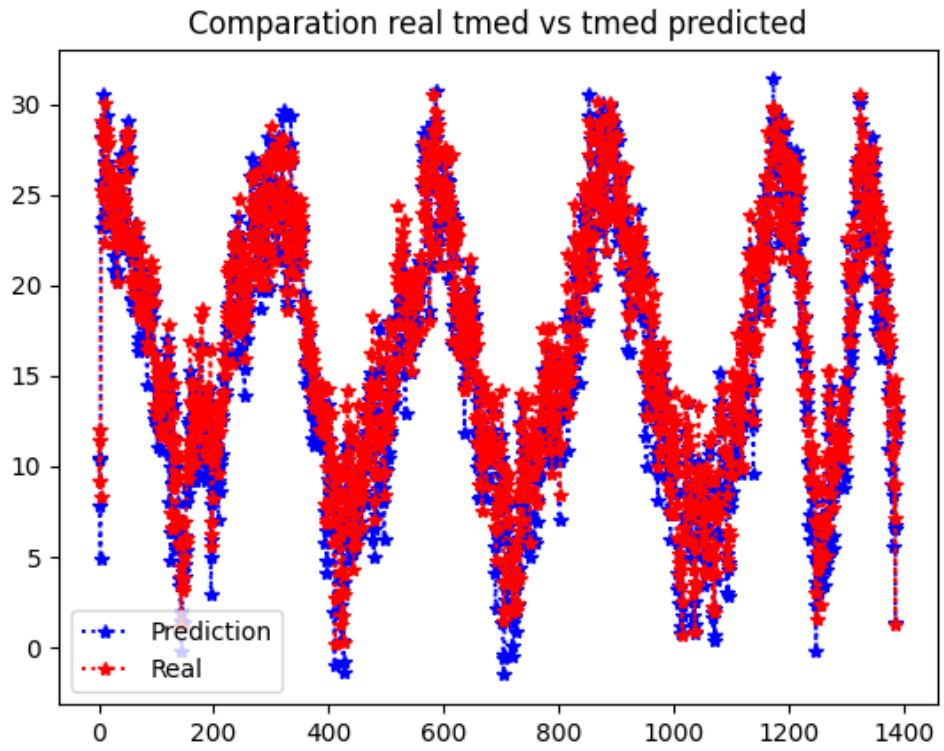

Figura 81: Gráfica de comparación de datos RNA con 10+1 días y 4 estaciones meteorológicas sin optimizar

	Original 10 days past final	Prediction 10 days past final
0	10.4	12.007149
1	10.5	9.230155
2	7.8	11.503784
3	4.9	8.328204
4	28.2	29.132857
...	...	...
1385	1.4	1.269459
1386	11.3	14.833441
1387	13.0	12.862412
1388	12.3	13.771837
1389	13.0	11.815707

[1390 rows x 2 columns]

Figura 82: Comparación de datos RNA con 10+1 días y 4 estaciones meteorológicas sin optimizar

RNA con 7+1 días, 4 estaciones meteorológicas y 300 de tamaño de lote

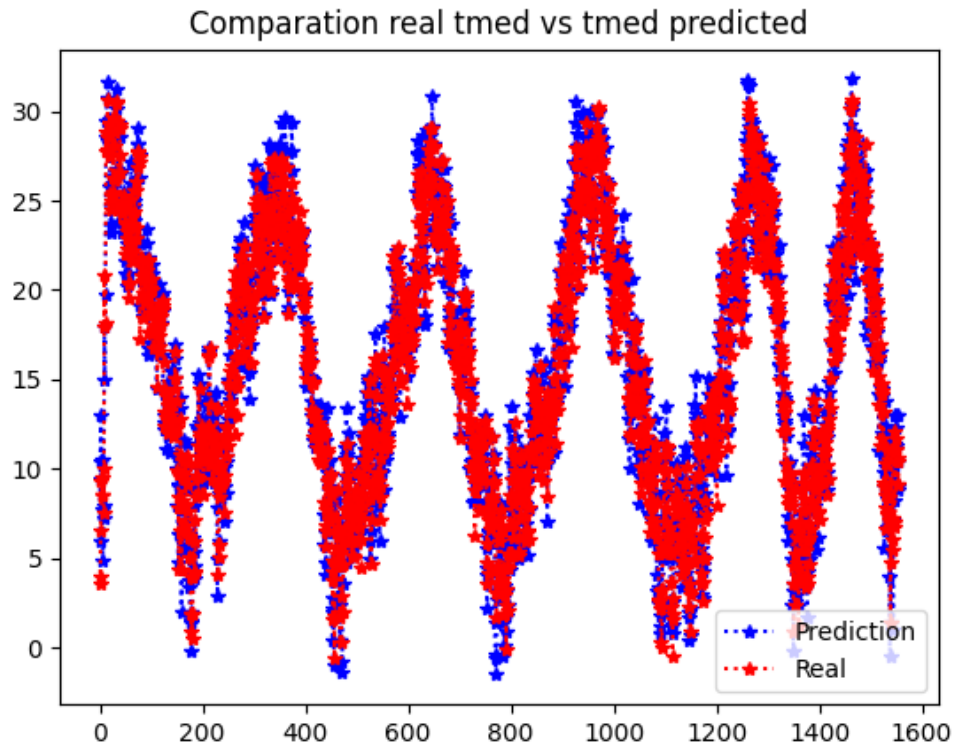

Figura 83: Gráfica de comparación de datos RNA con 7+1 días, 4 estaciones meteorológicas y 300 de tamaño de lote sin optimizar

	Original 7 days past final 300 batch	Prediction 7 days past final 300 batch
0	6.0	3.617858
1	9.4	4.029745
2	13.0	6.555136
3	10.4	9.334756
4	10.5	8.585066
...	...	...
1551	12.3	11.141723
1552	8.8	10.589971
1553	9.0	9.121849
1554	10.4	10.462486
1555	13.0	9.142177

[1556 rows x 2 columns]

Figura 84: Comparación de datos RNA con 7+1 días, 4 estaciones meteorológicas y 300 de tamaño de lote sin optimizar

Anexo D: Gráficos Q-Q de RNA sin optimizar

Las siguientes representaciones gráficas permiten observar cuan cerca está la distribución de un conjunto de dato a alguna distribución ideal.

RNA con 2+1 días y 1 estación meteorológica

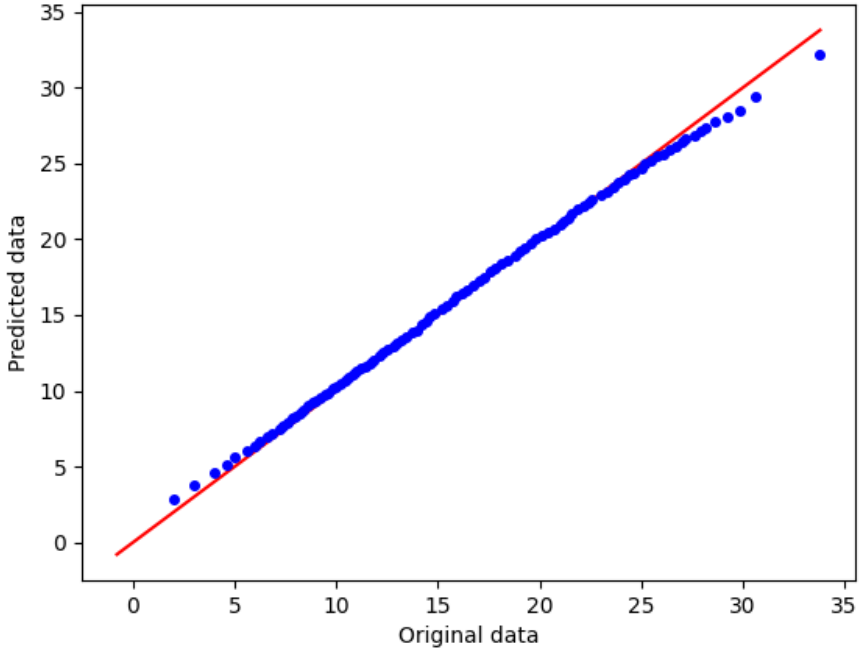

Figura 85: Gráfica Q-Q RNA con 2+1 días y 1 estación meteorológica sin optimizar

RNA con 5+1 días y 1 estación meteorológica

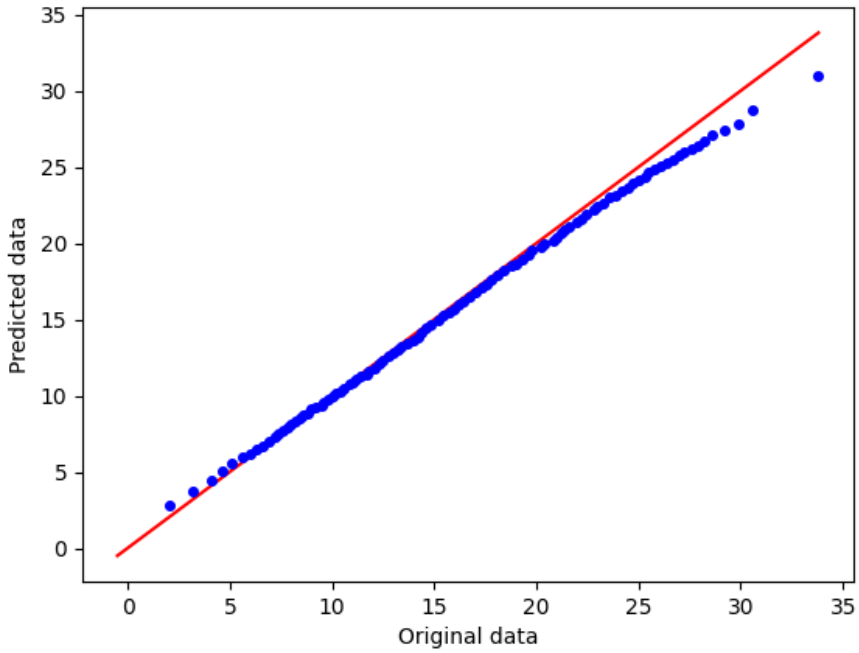

Figura 86: Gráfica Q-Q RNA con 5+1 días y 1 estación meteorológica sin optimizar

RNA con 7+1 días y 1 estación meteorológica

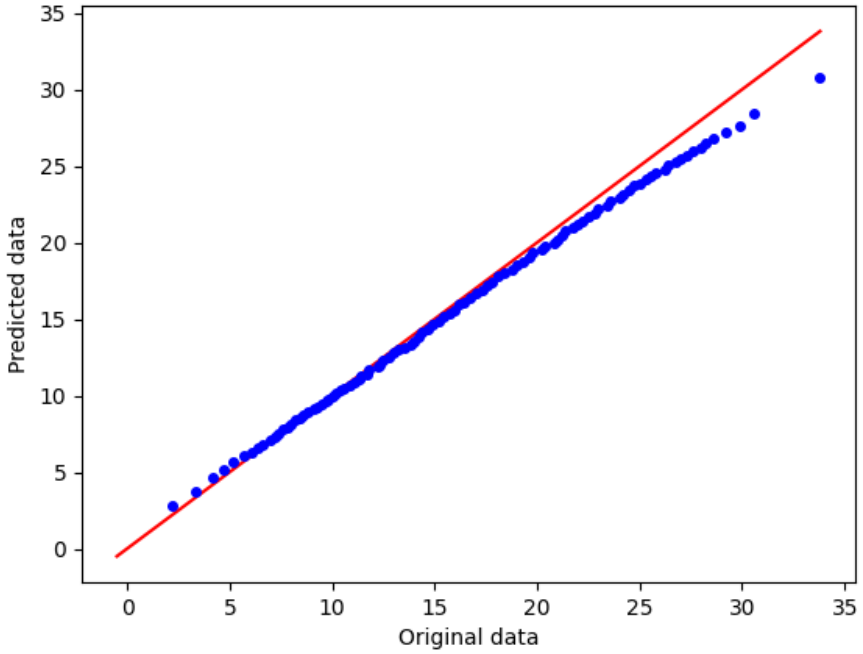

Figura 87: Gráfica Q-Q RNA con 7+1 días y 1 estación meteorológica sin optimizar

RNA con 10+1 días y 1 estación meteorológica

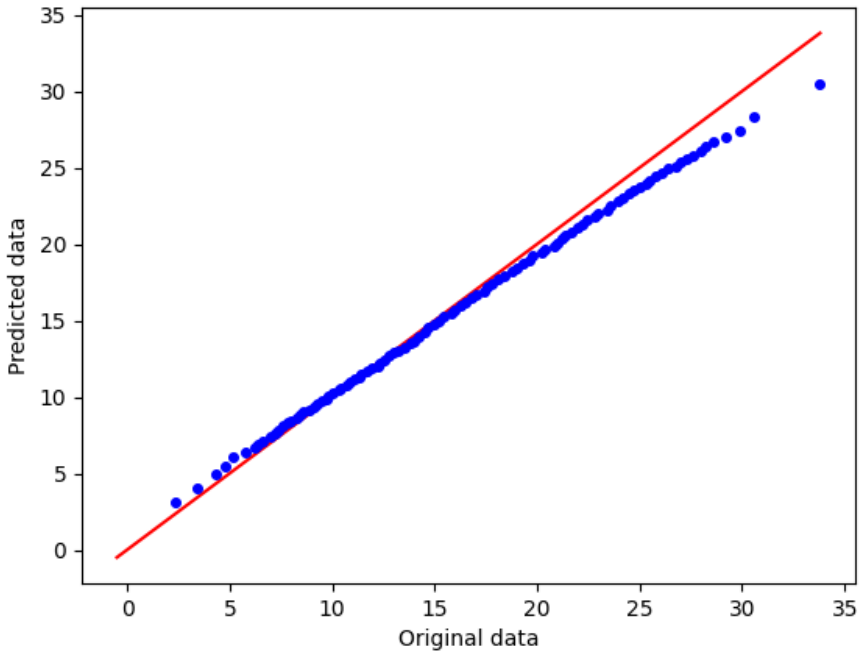

Figura 88: Gráfica Q-Q RNA con 10+1 días y 1 estación meteorológica sin optimizar

RNA con 2+1 días y 4 estaciones meteorológicas

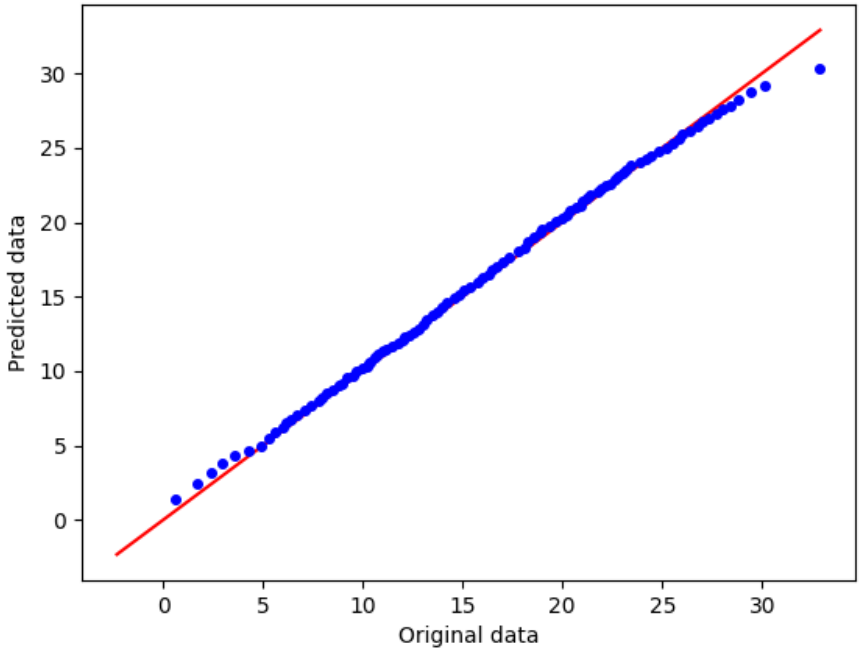

Figura 89: Gráfica Q-Q RNA con 2+1 días y 4 estaciones meteorológicas sin optimizar

RNA con 5+1 días y 4 estaciones meteorológicas

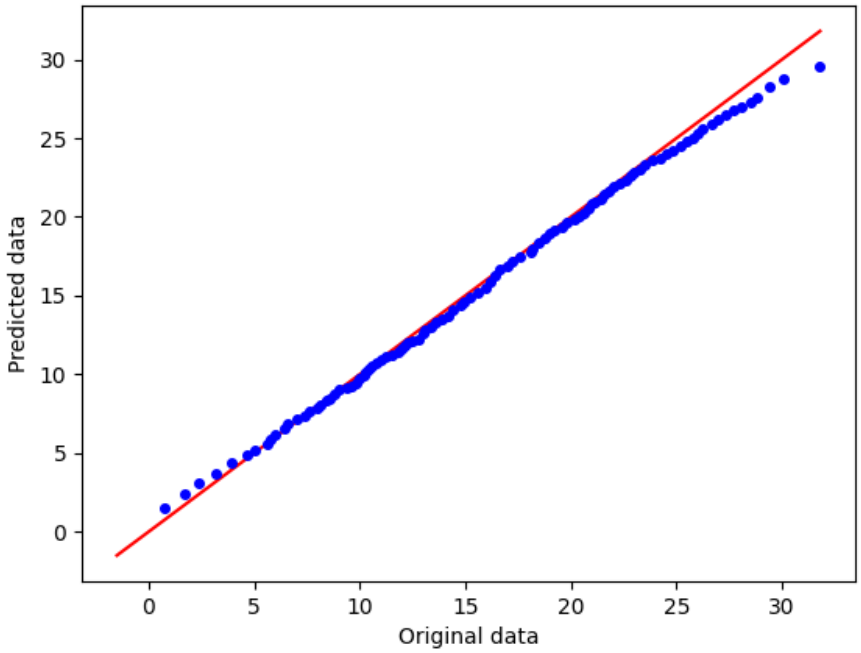

Figura 90: Gráfica Q-Q RNA con 5+1 días y 4 estaciones meteorológicas sin optimizar

RNA con 10+1 días y 4 estaciones meteorológicas

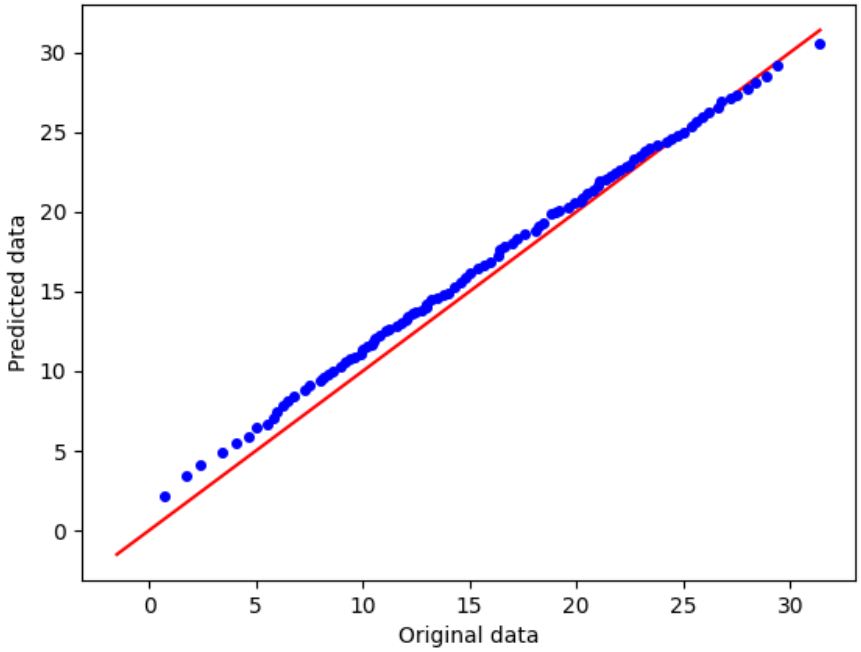

Figura 91: Gráfica Q-Q RNA con 10+1 días y 4 estaciones meteorológicas sin optimizar

RNA con 7+1 días, 4 estaciones meteorológicas y 300 de tamaño de lote

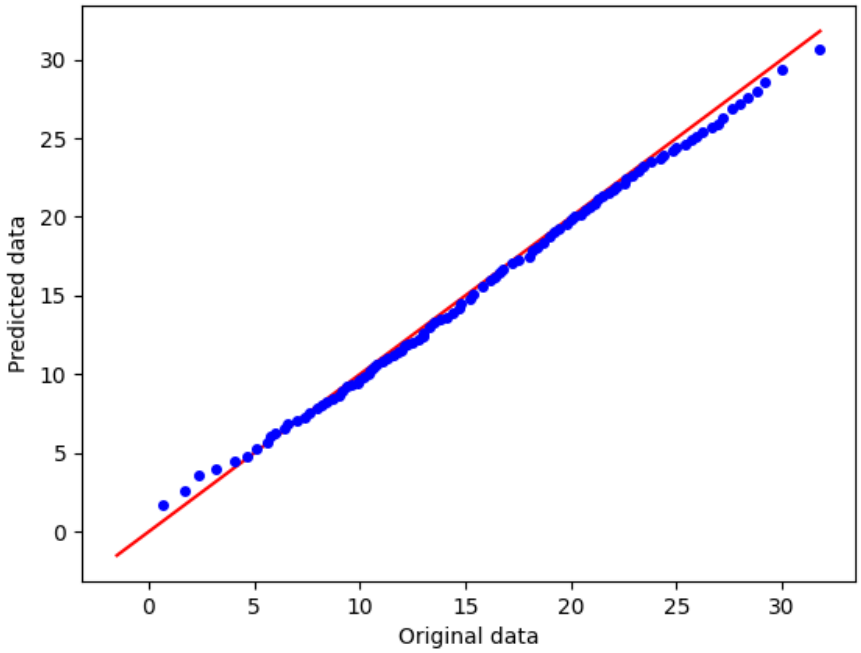

Figura 92: Gráfica Q-Q RNA con 7+1 días, 4 estaciones meteorológicas y 300 de tamaño de lote sin optimizar

Anexo E: MSE, RMSE, BIAS y Varianza de RNA sin optimizar

En esta sección se muestran los valores de: MSE, RMSE, BIAS y Varianza de cada RNA creada.

	RNA sin optimización
RNA con 2+1 días y 1 estación meteorológica	Mean Square Error: 3.569910603127335 Root Mean Square Error: 1.8894207056998542 BIAS: -0.07084053140663471 Variance: 3.6407511345339696
RNA con 5+1 días y 1 estación meteorológica	Mean Square Error: 3.6674061679326866 Root Mean Square Error: 1.9150473017481022 BIAS: 0.3854636321427307 Variance: 3.281942535789956
RNA con 7+1 días y 1 estación meteorológica	Mean Square Error: 3.8136507399631494 Root Mean Square Error: 1.9528570710533706 BIAS: 0.5019636302416188 Variance: 3.3116871097215306
RNA con 10+1 días y 1 estación meteorológica	Mean Square Error: 3.9098724529818867 Root Mean Square Error: 1.9773397414156948 BIAS: 0.45191869614648894 Variance: 3.4579537568353977
RNA con 2+1 días y 4 estaciones meteorológicas	Mean Square Error: 3.8382341850024906 Root Mean Square Error: 1.9591411855714969 BIAS: -0.1885808667456459 Variance: 4.0268150517481365
RNA con 5+1 días y 4 estaciones meteorológicas	Mean Square Error: 3.796256646444696 Root Mean Square Error: 1.9483984824580152 BIAS: 0.25455956094502774 Variance: 3.5416970854996683
RNA con 10+1 días y 4 estaciones meteorológicas	Mean Square Error: 4.863007732079059 Root Mean Square Error: 2.2052228304820036 BIAS: -0.9106031740955274 Variance: 5.773610906174587
RNA con 7+1 días, 4 estaciones meteorológicas y 300 de tamaño de lote	Mean Square Error: 4.7239738405889256 Root Mean Square Error: 2.173470460022157 BIAS: 0.2586969733515989 Variance: 4.465276867237327

Tabla 6: MSE, RMSE, BIAS y Varianza de RNAs sin optimizar

Anexo F: Estructura de las RNA optimizada

A continuación, se muestran las estructuras de los modelos de RNA creados haciendo uso de la optimización generada por Keras Tuner.

RNA con 2+1 días y 1 estación meteorológica

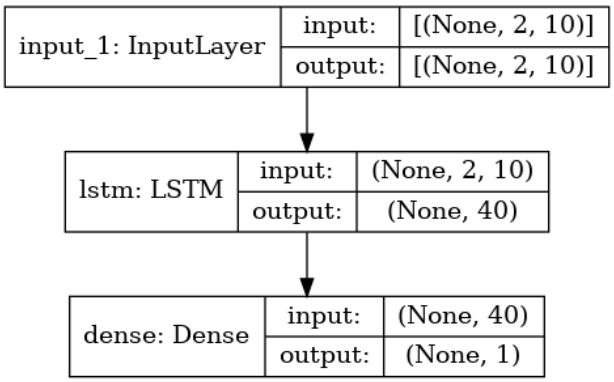

Figura 93: Estructura RNA con 2+1 días y 1 estación meteorológica optimizada

RNA con 5+1 días y 1 estación meteorológica

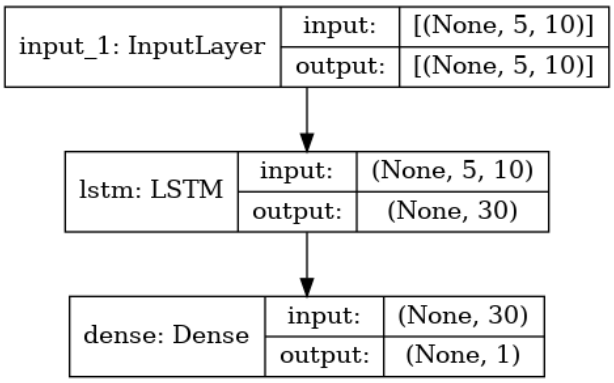

Figura 94: Estructura RNA con 5+1 días y 1 estación meteorológica optimizada

RNA con 7+1 días y 1 estación meteorológica

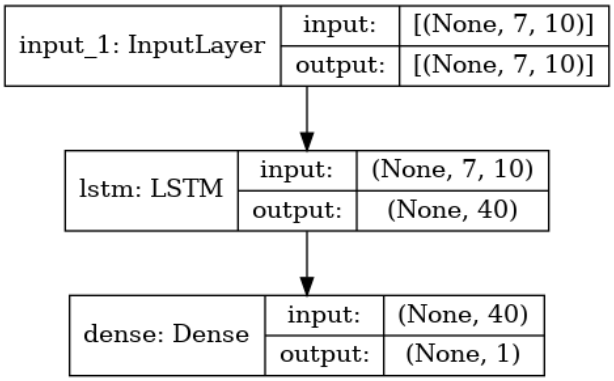

Figura 95: Estructura RNA con 7+1 días y 1 estación meteorológica optimizada

RNA con 10+1 días y 1 estación meteorológica

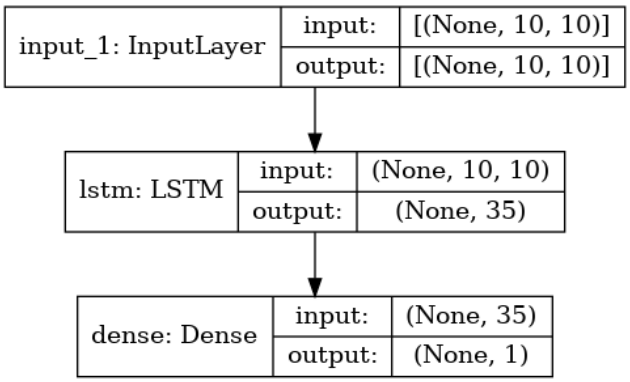

Figura 96: Estructura RNA con 10+1 días y 1 estación meteorológica optimizada

RNA con 2+1 días y 4 estaciones meteorológicas

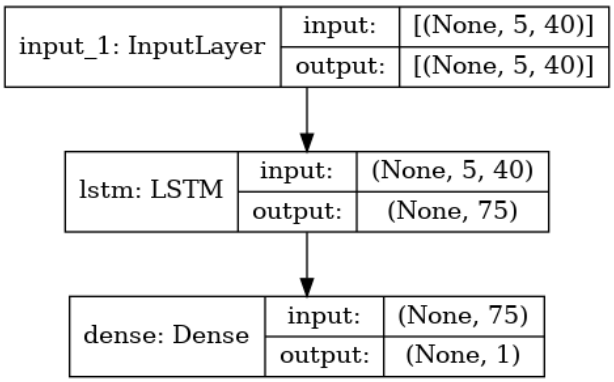

Figura 97: Estructura RNA con 2+1 días y 4 estaciones meteorológicas optimizada

RNA con 5+1 días y 4 estaciones meteorológicas

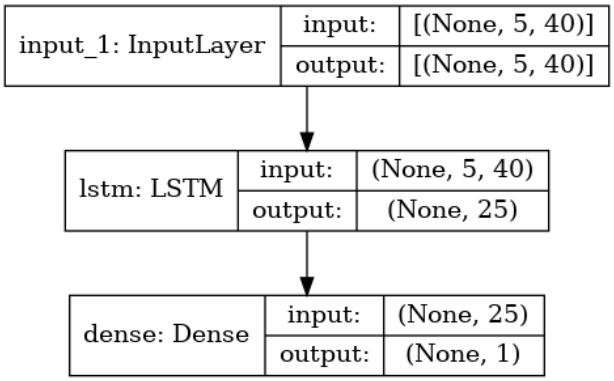

Figura 98: Estructura RNA con 5+1 días y 4 estaciones meteorológicas optimizada

RNA con 7+1 días y 4 estaciones meteorológicas

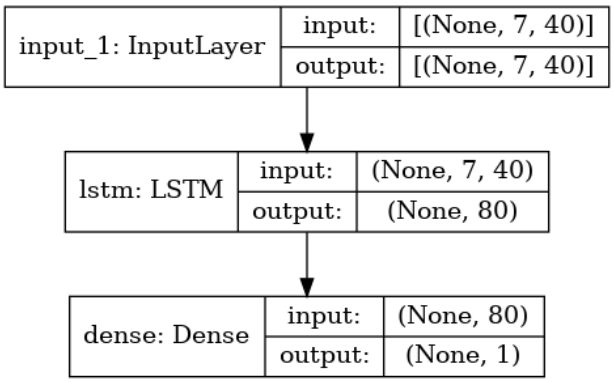

Figura 99: Estructura RNA con 7+1 días y 4 estaciones meteorológicas optimizada

RNA con 10+1 días y 4 estaciones meteorológicas

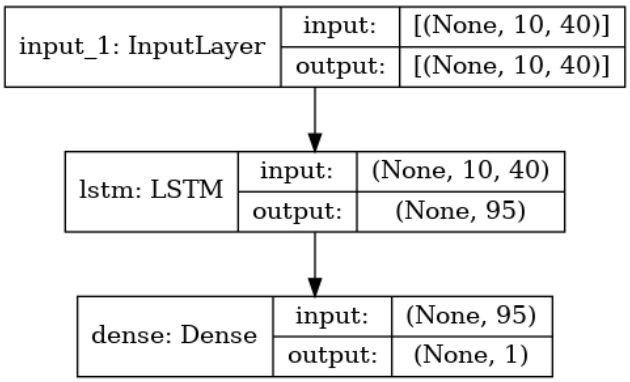

Figura 100: Estructura RNA con 10+1 días y 4 estaciones meteorológicas optimizada

RNA con 7+1 días, 4 estaciones meteorológicas y 300 de tamaño de lote

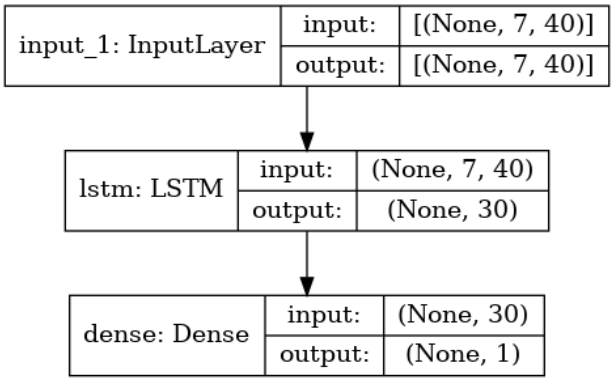

Figura 101: Estructura RNA con 7+1 días, 4 estaciones meteorológicas y 300 de tamaño de lote optimizada

Anexo G: Entrenamiento de las RNA optimizadas

En este punto se van a mostrar los entrenamientos de cada una de las RNA optimizadas y con las diferentes configuraciones que se han llevado a cabo.

RNA con 2+1 días y 1 estación meteorológica

```
Trial 5 Complete [00h 01m 09s]
val_loss: 0.0038858717307448387

Best val_loss So Far: 0.002864840983723601
Total elapsed time: 00h 05m 48s
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 2, 10)]           0
_________________________________________________________________
lstm (LSTM)                  (None, 40)                8160
_________________________________________________________________
dense (Dense)                (None, 1)                 41
=================================================================
Total params: 8,201
Trainable params: 8,201
Non-trainable params: 0
```

Figura 102: Entrenamiento RNA con 2+1 días y 1 estación meteorológica optimizada

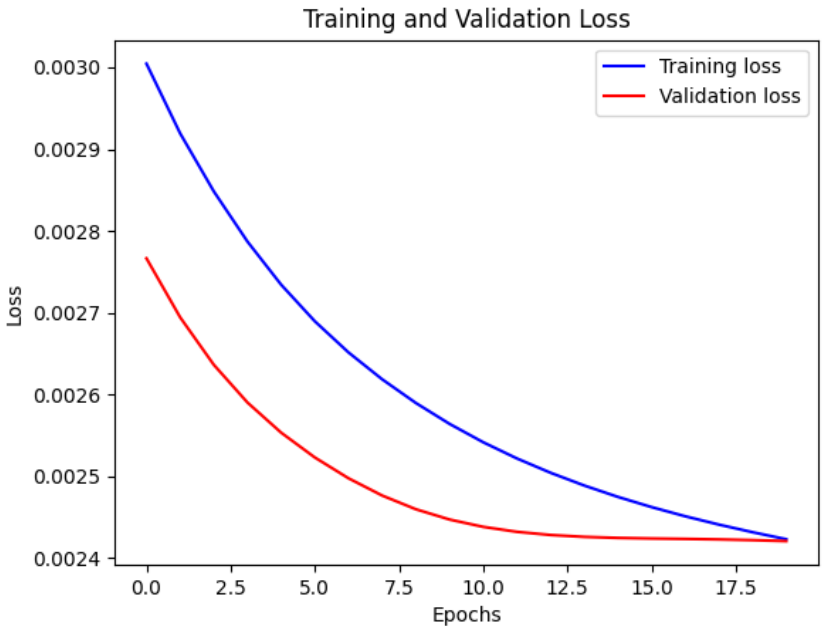

Figura 103: Gráfica de pérdida RNA con 2+1 días y 1 estación meteorológica optimizada

RNA con 5+1 días y 1 estación meteorológica

```
Trial 5 Complete [00h 01m 28s]
val_loss: 0.0030096106541653476

Best val_loss So Far: 0.0027304144265751043
Total elapsed time: 00h 07m 53s
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 5, 10)]           0
_________________________________________________________________
lstm (LSTM)                  (None, 30)                4920
_________________________________________________________________
dense (Dense)                (None, 1)                 31
=================================================================
Total params: 4,951
Trainable params: 4,951
Non-trainable params: 0
```

Figura 104: Entrenamiento RNA con 5+1 días y 1 estación meteorológica optimizada

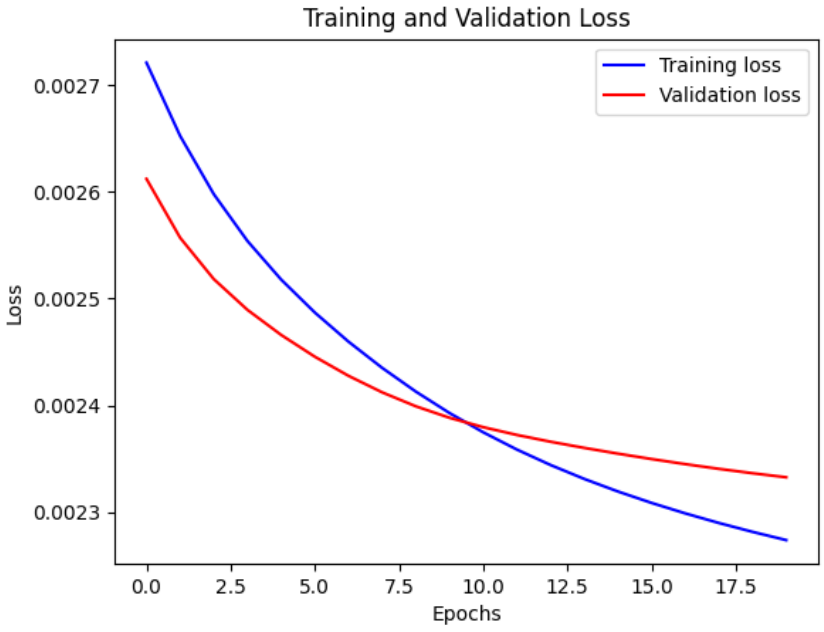

Figura 105: Gráfica de pérdida RNA con 5+1 días y 1 estación meteorológica optimizada

RNA con 7+1 días y 1 estación meteorológica

```
Trial 5 Complete [00h 01m 44s]
val_loss: 0.003573587785164515

Best val_loss So Far: 0.003005419780189792
Total elapsed time: 00h 08m 25s
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 7, 10)]           0
_________________________________________________________________
lstm (LSTM)                  (None, 40)                8160
_________________________________________________________________
dense (Dense)                (None, 1)                 41
=================================================================
Total params: 8,201
Trainable params: 8,201
Non-trainable params: 0
```

Figura 106: Entrenamiento RNA con 7+1 días y 1 estación meteorológica optimizada

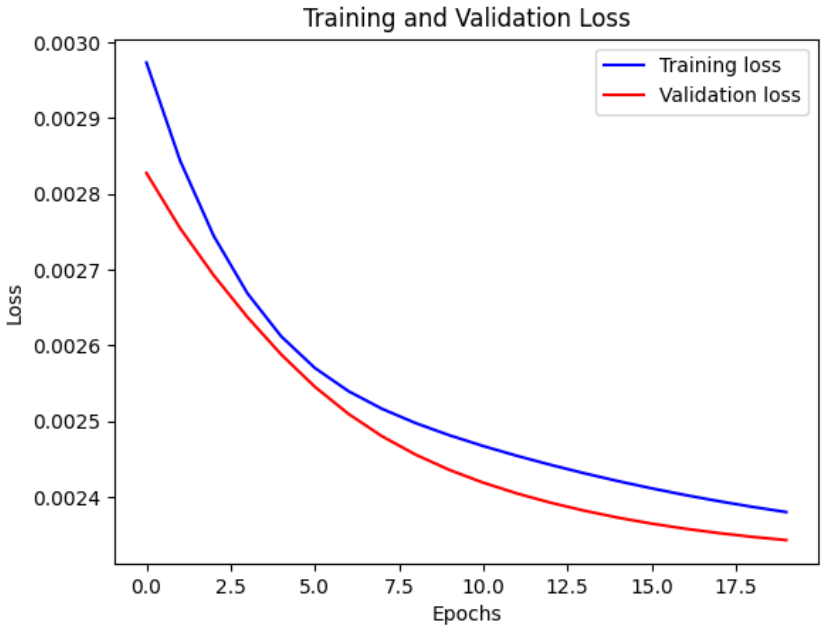

Figura 107: Gráfica de pérdida RNA con 7+1 días y 1 estación meteorológica optimizada

RNA con 10+1 días y 1 estación meteorológica

```
Trial 5 Complete [00h 02m 05s]
val_loss: 0.0036328445809582868

Best val_loss So Far: 0.0028039168876906237
Total elapsed time: 00h 09m 54s
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 10, 10)]          0
_________________________________________________________________
lstm (LSTM)                  (None, 35)                6440
_________________________________________________________________
dense (Dense)                (None, 1)                 36
=================================================================
Total params: 6,476
Trainable params: 6,476
Non-trainable params: 0
```

Figura 108: Entrenamiento RNA con 10+1 días y 1 estación meteorológica optimizada

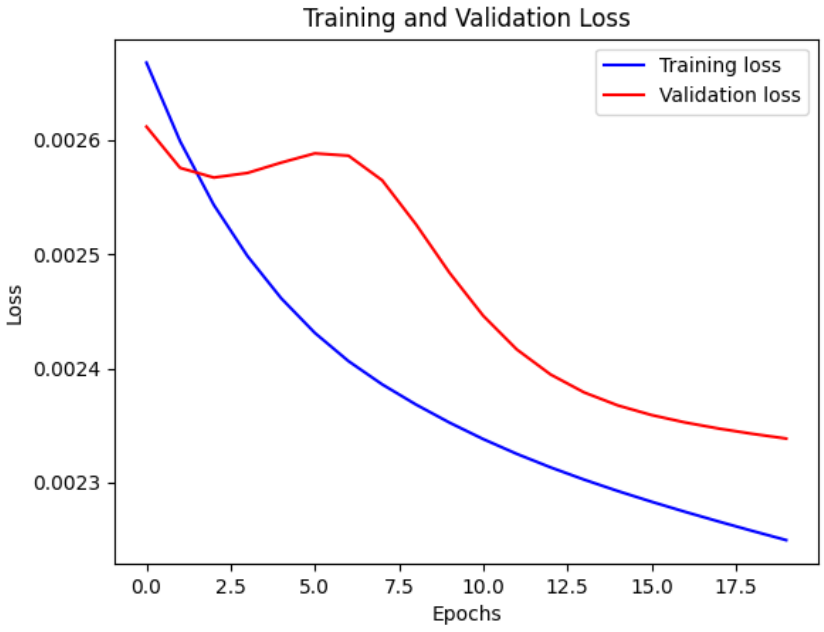

Figura 109: Gráfica de pérdida RNA con 10+1 días y 1 estación meteorológica optimizada

RNA con 2+1 días y 4 estaciones meteorológicas

```
Trial 5 Complete [00h 00m 41s]
val_loss: 0.003016539771730701

Best val_loss So Far: 0.003016539771730701
Total elapsed time: 00h 03m 27s
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 2, 40)]           0
_________________________________________________________________
lstm (LSTM)                  (None, 75)                34800
_________________________________________________________________
dense (Dense)                (None, 1)                 76
=================================================================
Total params: 34,876
Trainable params: 34,876
Non-trainable params: 0
```

Figura 110: Entrenamiento RNA con 2+1 días y 4 estaciones meteorológicas optimizada

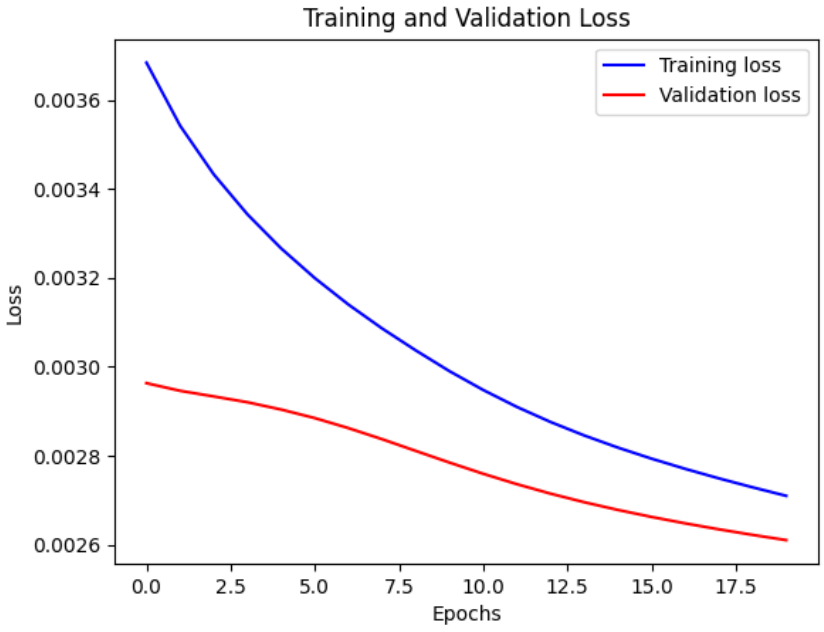

Figura 111: Gráfica de pérdida RNA con 2+1 días y 4 estaciones meteorológicas optimizada

RNA con 5+1 días y 4 estaciones meteorológicas

```
Trial 5 Complete [00h 00m 37s]
val_loss: 0.0048651364631950855

Best val_loss So Far: 0.003163048687080542
Total elapsed time: 00h 03m 18s
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 5, 40)]           0
_________________________________________________________________
lstm (LSTM)                  (None, 25)                6600
_________________________________________________________________
dense (Dense)                (None, 1)                 26
=================================================================
Total params: 6,626
Trainable params: 6,626
Non-trainable params: 0
```

Figura 112: Entrenamiento RNA con 5+1 días y 4 estaciones meteorológicas optimizada

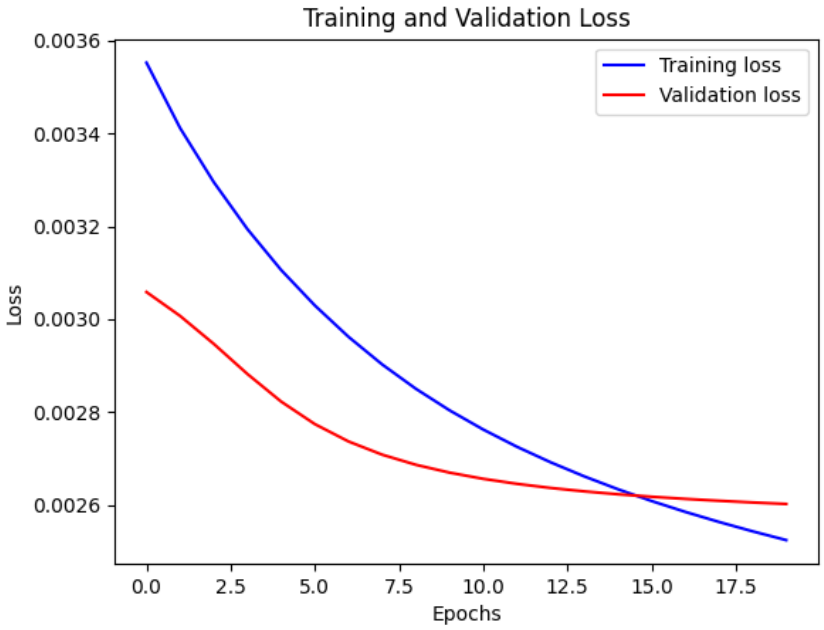

Figura 113: Gráfica de pérdida RNA con 5+1 días y 4 estaciones meteorológicas optimizada

RNA con 7+1 días y 4 estaciones meteorológicas

```
Trial 5 Complete [00h 00m 40s]
val_loss: 0.0040112079586833715

Best val_loss So Far: 0.0031978539967288575
Total elapsed time: 00h 03m 21s
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 7, 40)]           0
_________________________________________________________________
lstm (LSTM)                  (None, 80)                38720
_________________________________________________________________
dense (Dense)                (None, 1)                 81
=================================================================
Total params: 38,801
Trainable params: 38,801
Non-trainable params: 0
```

Figura 114: Entrenamiento RNA con 7+1 días y 4 estaciones meteorológicas optimizada

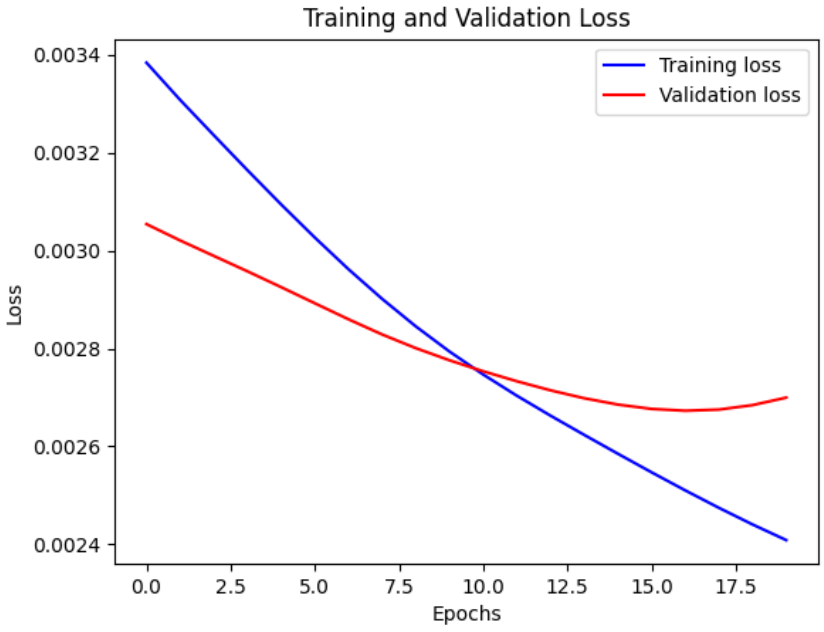

Figura 115: Gráfica de pérdida RNA con 7+1 días y 4 estaciones meteorológicas optimizada

RNA con 10+1 días y 4 estaciones meteorológicas

```
Trial 5 Complete [00h 00m 47s]
val_loss: 0.0032194660355647406

Best val_loss So Far: 0.0032194660355647406
Total elapsed time: 00h 03m 22s
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 10, 40)]          0
_________________________________________________________________
lstm (LSTM)                  (None, 95)                51680
_________________________________________________________________
dense (Dense)                (None, 1)                 96
=================================================================
Total params: 51,776
Trainable params: 51,776
Non-trainable params: 0
```

Figura 116: Entrenamiento RNA con 10+1 días y 4 estaciones meteorológicas optimizada

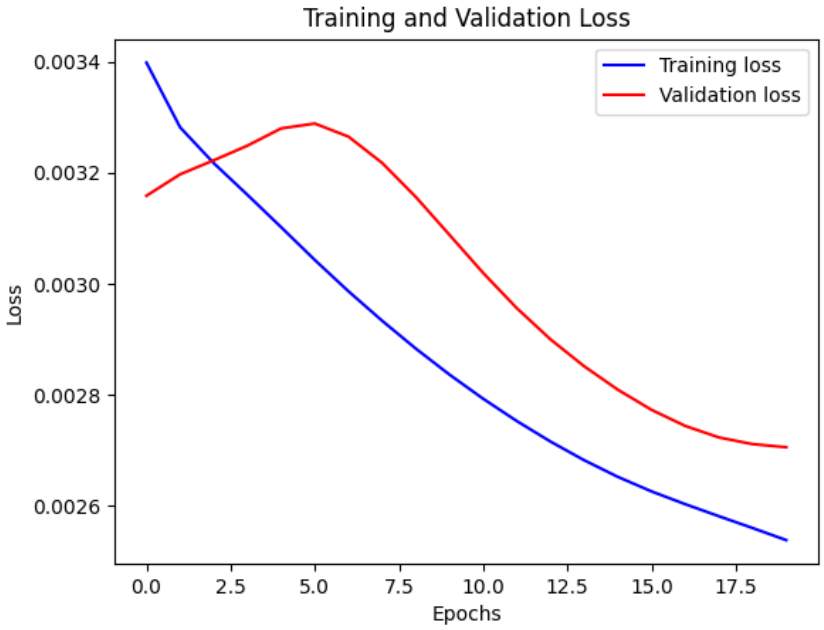

Figura 117: Gráfica de pérdida RNA con 10+1 días y 4 estaciones meteorológicas optimizada

RNA con 7+1 días, 4 estaciones meteorológicas y 300 de tamaño de lote

```
Trial 5 Complete [00h 00m 09s]
val_loss: 0.0183229794104894

Best val_loss So Far: 0.0058437068946659565
Total elapsed time: 00h 00m 43s
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 7, 40)]           0
_________________________________________________________________
lstm (LSTM)                  (None, 30)                8520
_________________________________________________________________
dense (Dense)                (None, 1)                 31
=================================================================
Total params: 8,551
Trainable params: 8,551
Non-trainable params: 0
```

Figura 118: Entrenamiento RNA con 7+1 días, 4 estaciones meteorológicas y 300 de tamaño de lote optimizada

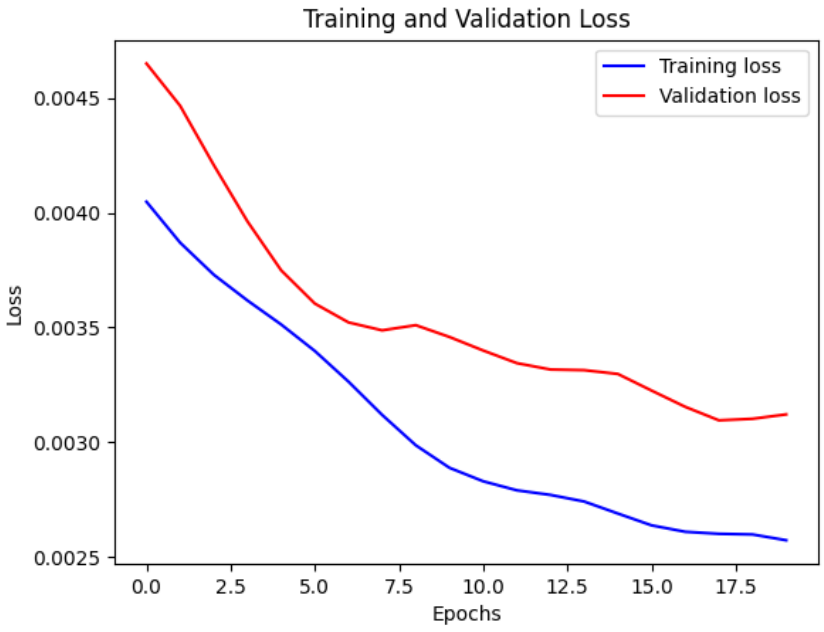

Figura 119: Gráfica de pérdida RNA con 7+1 días, 4 estaciones meteorológicas y 300 de tamaño de lote optimizada

Anexo H: Comparación datos reales y datos predichos de las RNA optimizadas

En esta sección se van a mostrar las diferentes gráficas donde se comparan los datos reales de las temperaturas medias con las temperaturas predichas por cada RNA creada.

RNA con 2+1 días y 1 estación meteorológica

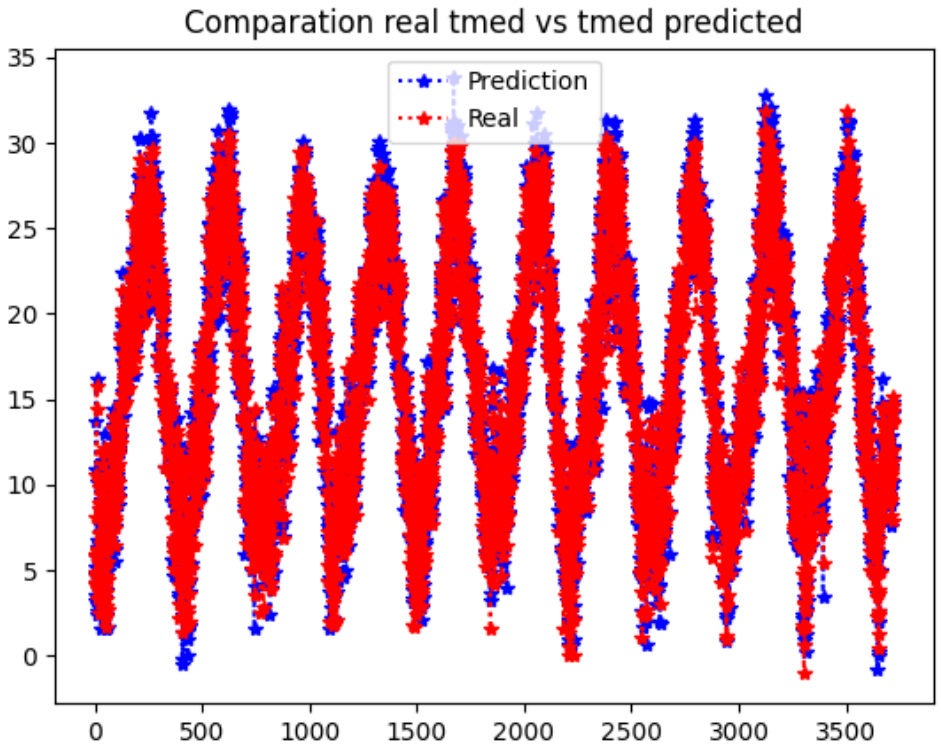

Figura 120: Gráfica de comparación de datos RNA con 2+1 días y 1 estación meteorológica optimizada

	Original	Prediction
0	3.2	4.308866
1	5.8	5.879933
2	4.8	5.931552
3	4.3	4.929691
4	3.5	4.689928
...	...	...
3717	12.3	13.746653
3718	14.6	14.604184
3719	14.9	14.338325
3720	14.2	15.134861
3721	14.2	13.639708

[3722 rows x 2 columns]

Figura 121: Comparación de datos RNA con 2+1 días y 1 estación meteorológica optimizada

RNA con 5+1 días y 1 estación meteorológica

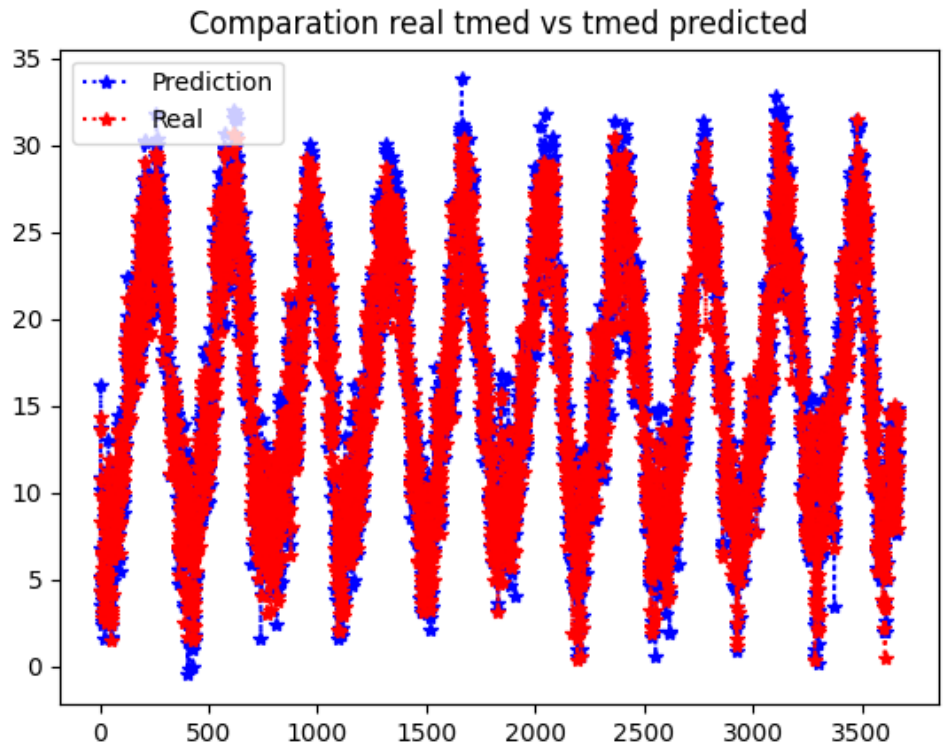

Figura 122: Gráfica de comparación de datos RNA con 5+1 días y 1 estación meteorológica optimizada

	Original	Prediction
0	4.3	5.063610
1	3.5	4.289874
2	5.2	4.322696
3	10.9	8.320785
4	13.7	10.781354
...	...	...
3666	12.3	13.040918
3667	14.6	14.228423
3668	14.9	14.436423
3669	14.2	14.658599
3670	14.2	13.583238

[3671 rows x 2 columns]

Figura 123: Comparación de datos RNA con 5+1 días y 1 estación meteorológica optimizada

RNA con 7+1 días y 1 estación meteorológica

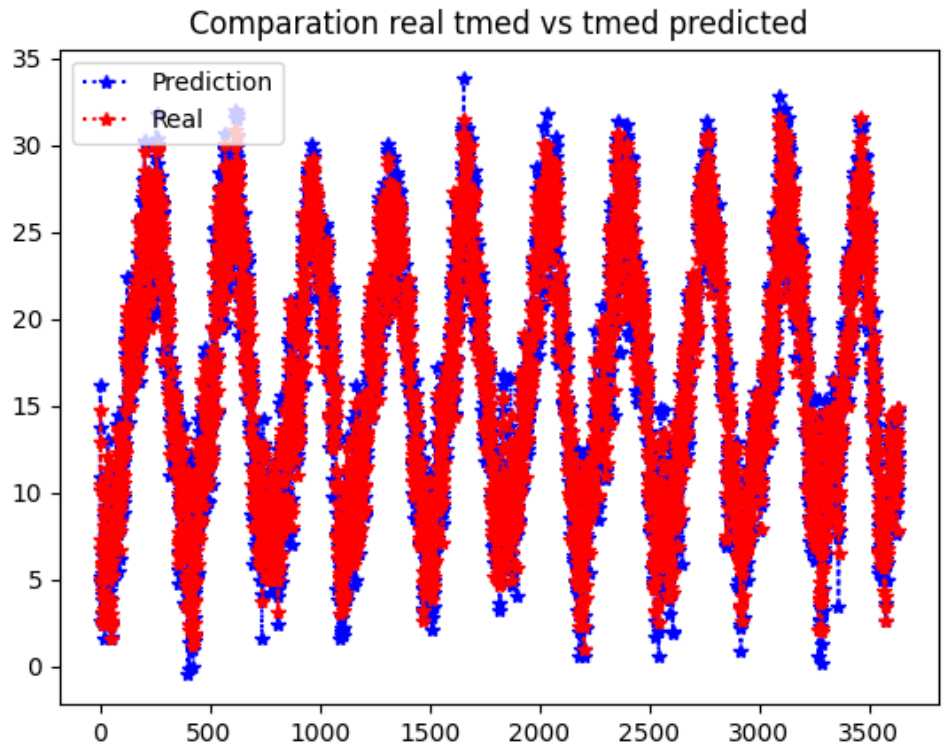

Figura 124: Gráfica de comparación de datos RNA con 7+1 días y 1 estación meteorológica optimizada

	Original	Prediction
0	5.2	4.966565
1	10.9	6.765772
2	13.7	10.483481
3	16.2	12.930975
4	10.2	14.760623
...	...	...
3632	12.3	12.960071
3633	14.6	13.464449
3634	14.9	14.863098
3635	14.2	14.861264
3636	14.2	14.271858

[3637 rows x 2 columns]

Figura 125: Comparación de datos RNA con 7+1 días y 1 estación meteorológica optimizada

RNA con 10+1 días y 1 estación meteorológica

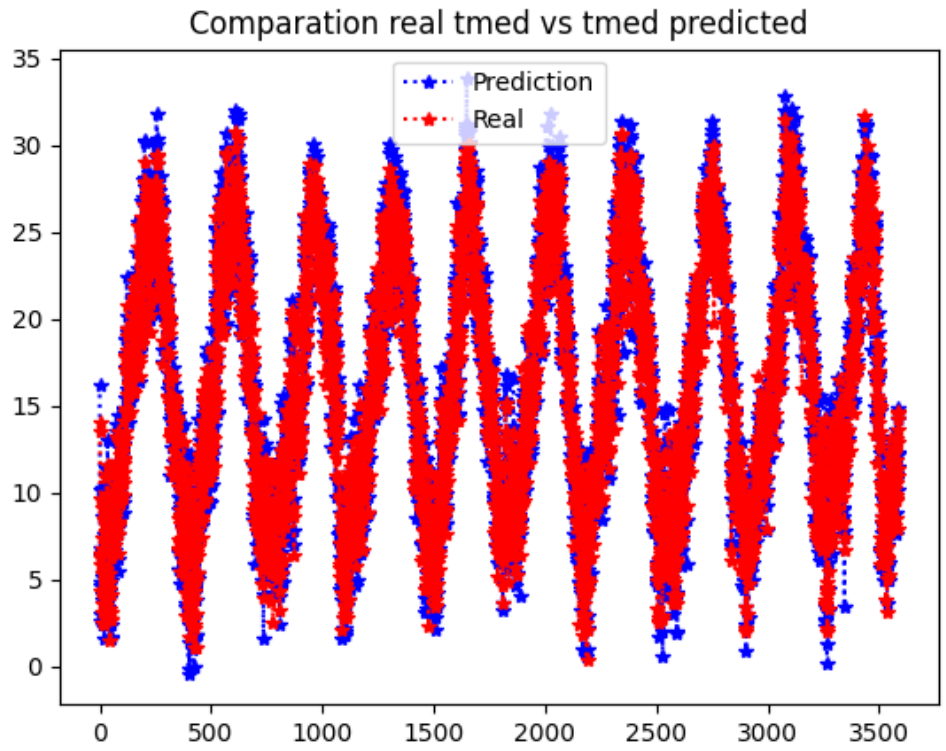

Figura 126: Gráfica de comparación de datos RNA con 10+1 días y 1 estación meteorológica optimizada

	Original	Prediction
0	16.2	13.506988
1	10.2	14.064401
2	6.8	9.637535
3	2.6	6.296546
4	4.8	4.342975
...	...	...
3584	12.3	12.891968
3585	14.6	14.248703
3586	14.9	14.499906
3587	14.2	14.662041
3588	14.2	13.656287

[3589 rows x 2 columns]

Figura 127: Comparación de datos RNA con 10+1 días y 1 estación meteorológica optimizada

RNA con 2+1 días y 4 estaciones meteorológicas

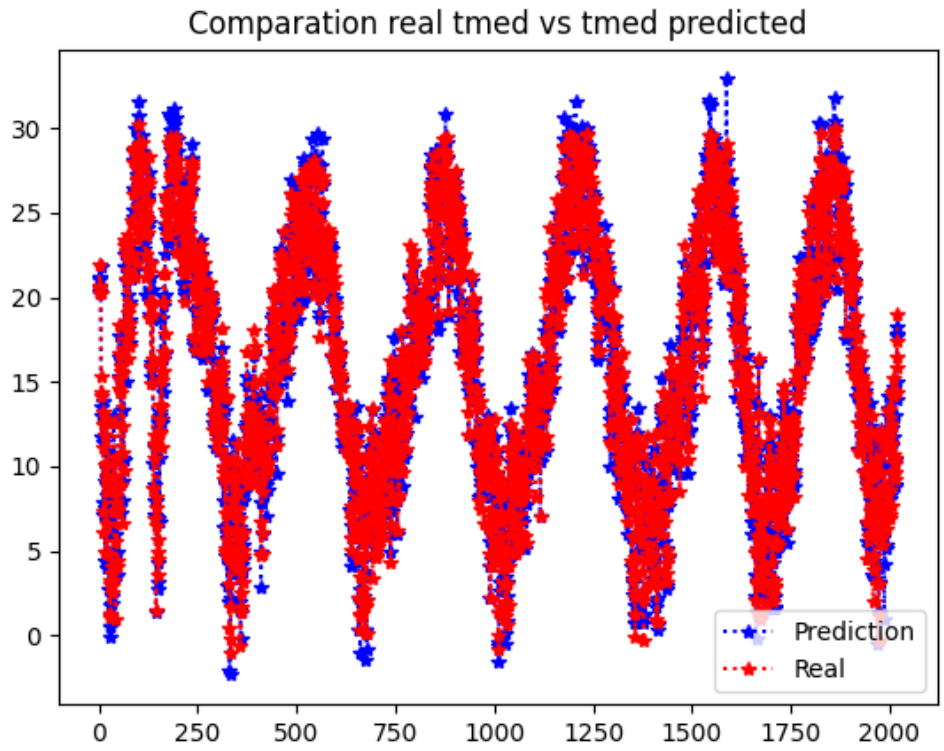

Figura 128: Gráfica de comparación de datos RNA con 2+1 días y 4 estaciones meteorológicas optimizada

	Original 2 days past final	Prediction 2 days past KT final
0	21.1	20.664664
1	21.2	20.592507
2	20.4	21.986205
3	21.1	20.543837
4	20.3	20.304185
...	...	...
2016	14.0	14.426941
2017	15.0	15.817044
2018	17.8	16.474937
2019	18.3	17.398363
2020	17.9	18.989412

[2021 rows x 2 columns]

Figura 129: Comparación de datos RNA con 2+1 días y 4 estaciones meteorológicas optimizada

RNA con 5+1 días y 4 estaciones meteorológicas

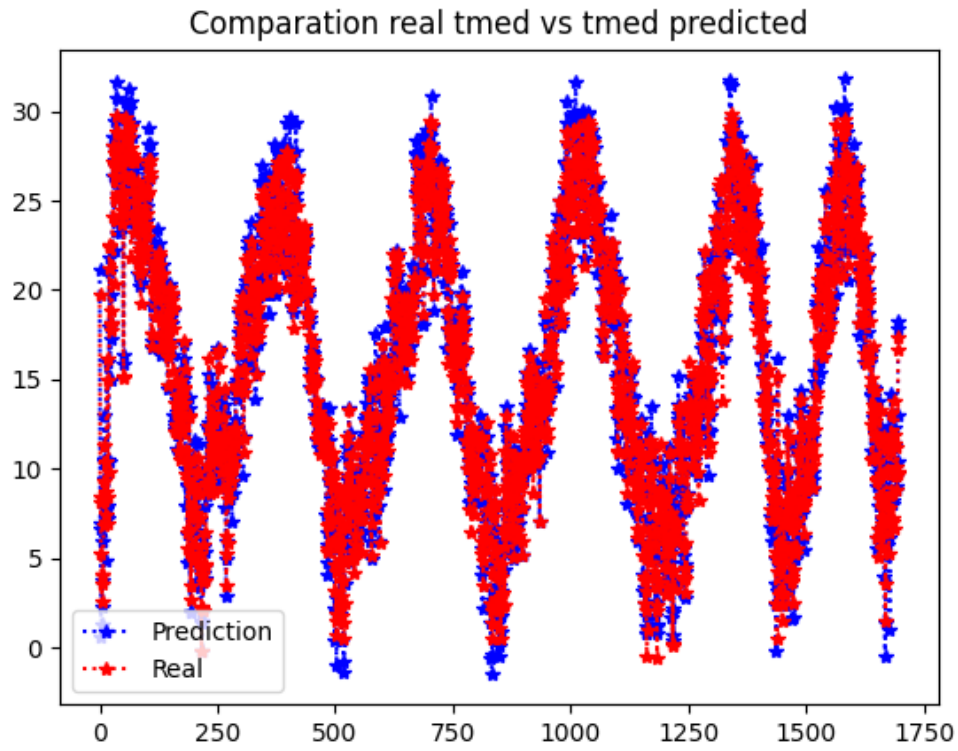

Figura 130: Gráfica de comparación de datos RNA con 5+1 días y 4 estaciones meteorológicas optimizada

	Original 5 days past final	Prediction 5 days past KT final
0	21.1	19.739476
1	7.0	8.417008
2	6.7	8.134650
3	0.6	5.300205
4	1.3	2.595953
...	...	...
1693	9.0	8.857956
1694	10.4	9.708953
1695	13.0	9.865788
1696	18.3	16.716599
1697	17.9	17.330999

[1698 rows x 2 columns]

Figura 131: Comparación de datos RNA con 5+1 días y 4 estaciones meteorológicas optimizada

RNA con 7+1 días y 4 estaciones meteorológicas

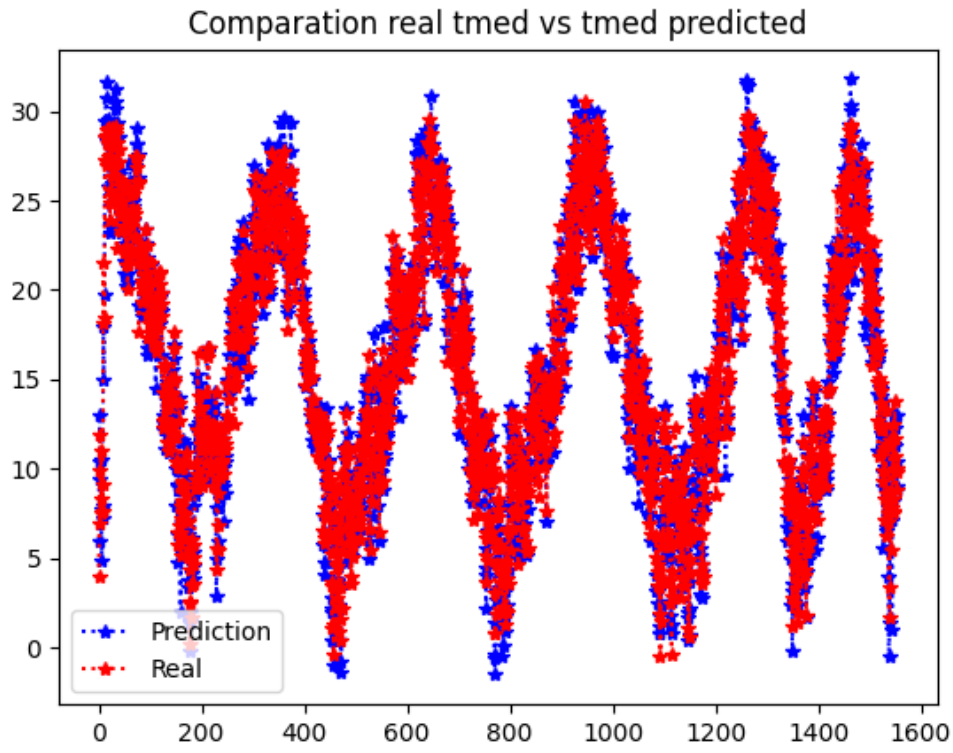

Figura 132: Gráfica de comparación de datos RNA con 7+1 días y 4 estaciones meteorológicas optimizada

	Original 7 days past final	Prediction 7 days past KT final
0	6.0	3.963159
1	9.4	6.929933
2	13.0	9.910899
3	10.4	11.891645
4	10.5	8.315996
...	...	...
1551	12.3	10.703299
1552	8.8	9.873938
1553	9.0	8.879534
1554	10.4	10.065097
1555	13.0	10.013249

[1556 rows x 2 columns]

Figura 133: Comparación de datos RNA con 7+1 días y 4 estaciones meteorológicas optimizada

RNA con 10+1 días y 4 estaciones meteorológicas

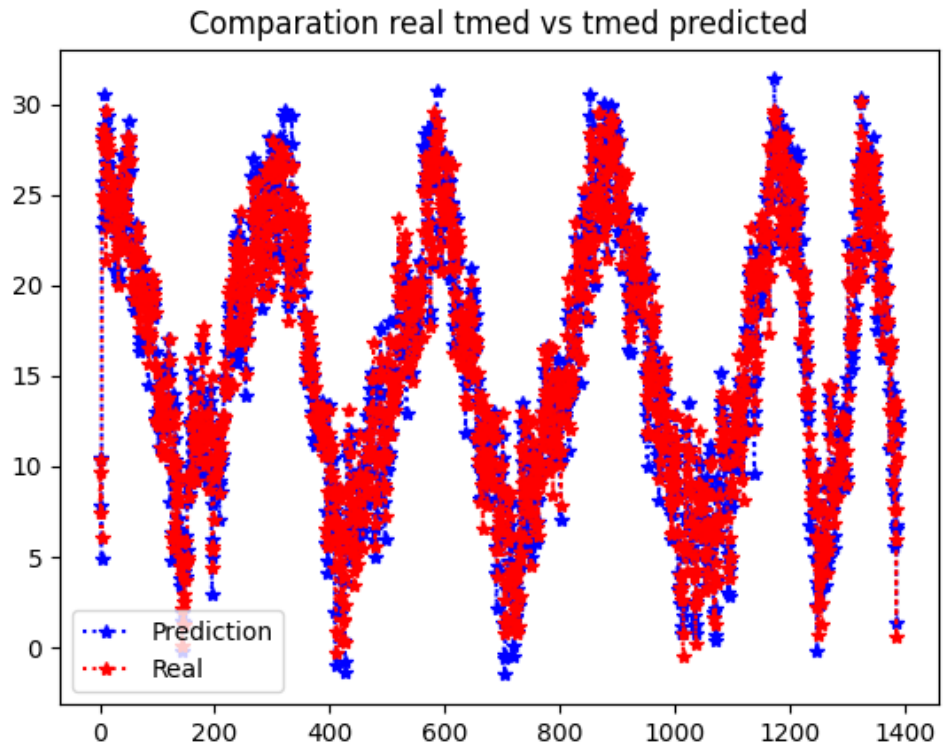

Figura 134: Gráfica de comparación de datos RNA con 10+1 días y 4 estaciones meteorológicas optimizada

	Original 10 days past final	Prediction 10 days past KT final
0	10.4	10.415373
1	10.5	7.472040
2	7.8	9.700851
3	4.9	6.124799
4	28.2	28.577800
...	...	...
1385	1.4	0.617797
1386	11.3	13.648754
1387	13.0	12.037145
1388	12.3	11.927185
1389	13.0	10.473046

[1390 rows x 2 columns]

Figura 135: Comparación de datos RNA con 10+1 días y 4 estaciones meteorológicas optimizada

RNA con 7+1 días, 4 estaciones meteorológicas y 300 de tamaño de lote

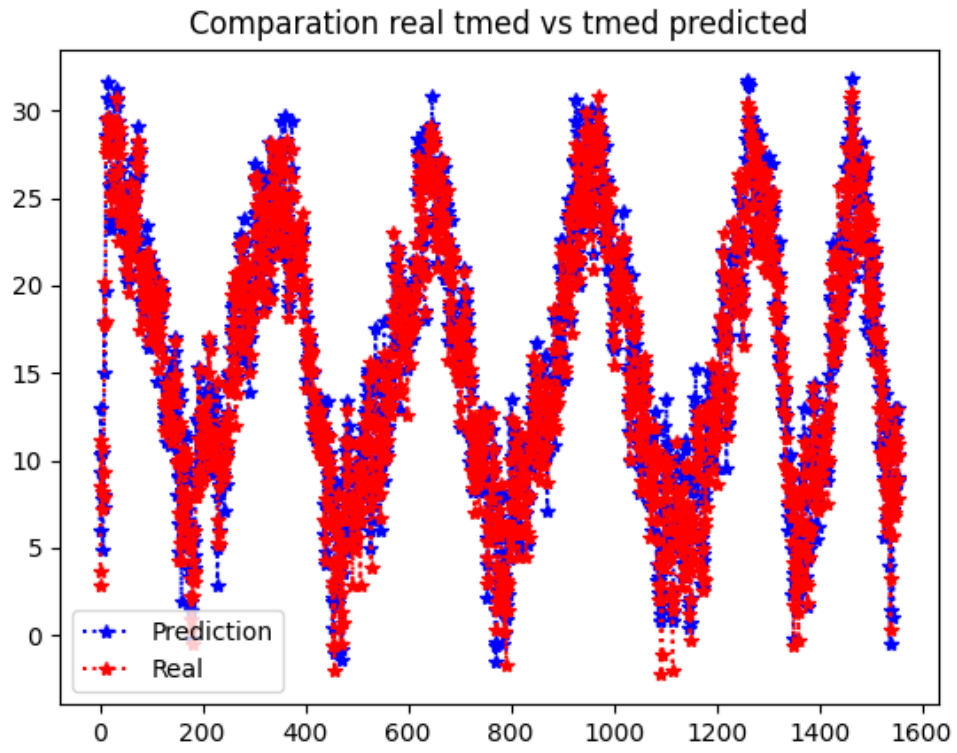

Figura 136: Gráfica de comparación de datos RNA con 7+1 días, 4 estaciones meteorológicas y 300 de tamaño de lote optimizada

	Original 7 days past final 300 batch	Prediction 7 days past KT final 300 batch
0	6.0	2.845706
1	9.4	3.635357
2	13.0	8.580562
3	10.4	11.172212
4	10.5	8.212869
...	...	...
1551	12.3	10.335985
1552	8.8	11.010354
1553	9.0	8.983756
1554	10.4	10.240558
1555	13.0	7.829653

[1556 rows x 2 columns]

Figura 137: Comparación de datos RNA con 7+1 días, 4 estaciones meteorológicas y 300 de tamaño de lote optimizada

Anexo I: Gráficos Q-Q de las RNA optimizadas

Las siguientes representaciones gráficas permiten observar cuan cerca está la distribución de un conjunto de dato a alguna distribución ideal.

RNA con 2+1 días y 1 estación meteorológica

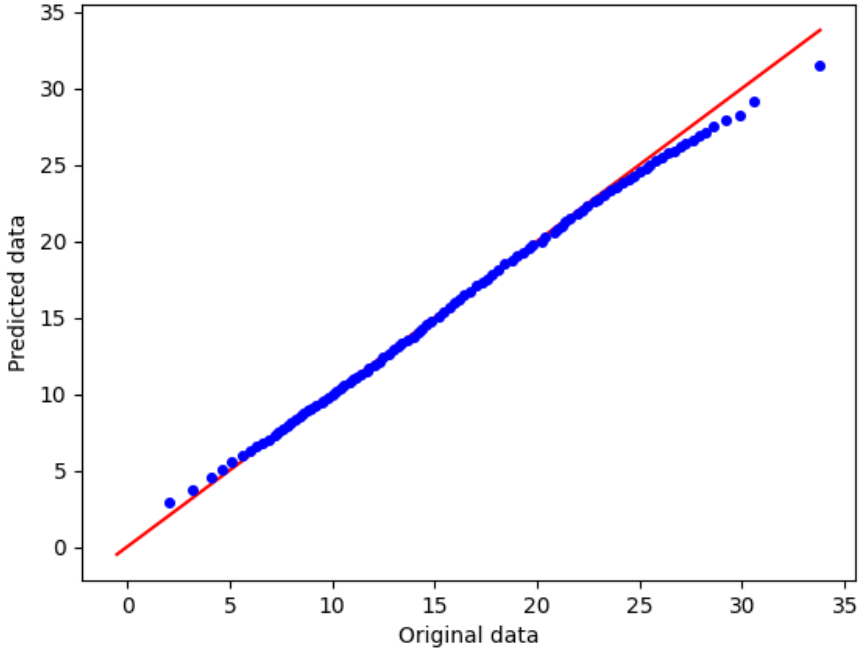

Figura 138: Gráfica Q-Q RNA con 2+1 días y 1 estación meteorológica optimizada

RNA con 5+1 días y 1 estación meteorológica

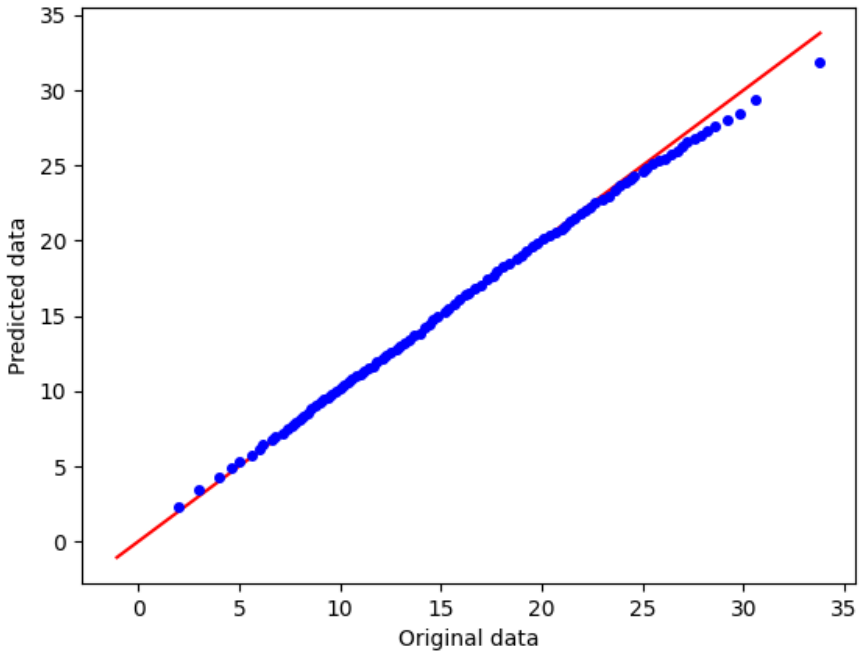

Figura 139: Gráfica Q-Q RNA con 5+1 días y 1 estación meteorológica optimizada

RNA con 7+1 días y 1 estación meteorológica

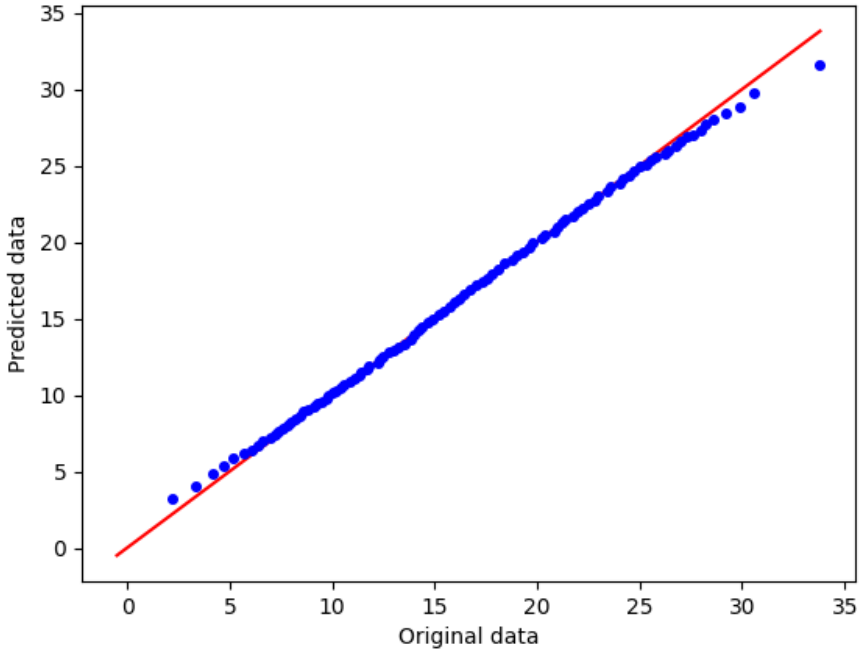

Figura 140: Gráfica Q-Q RNA con 7+1 días y 1 estación meteorológica optimizada

RNA con 10+1 días y 1 estación meteorológica

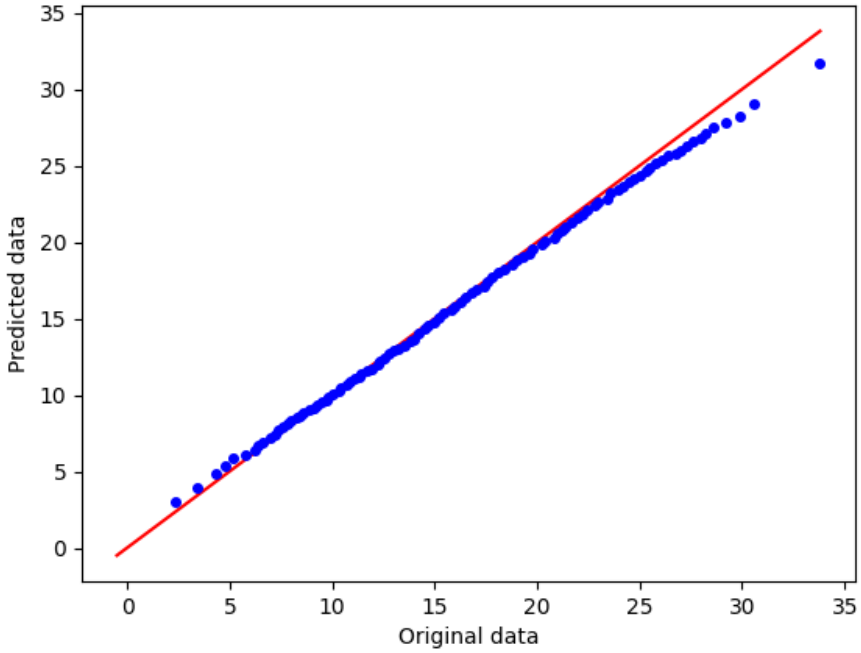

Figura 141: Gráfica Q-Q RNA con 10+1 días y 1 estación meteorológica optimizada

RNA con 2+1 días y 4 estaciones meteorológicas

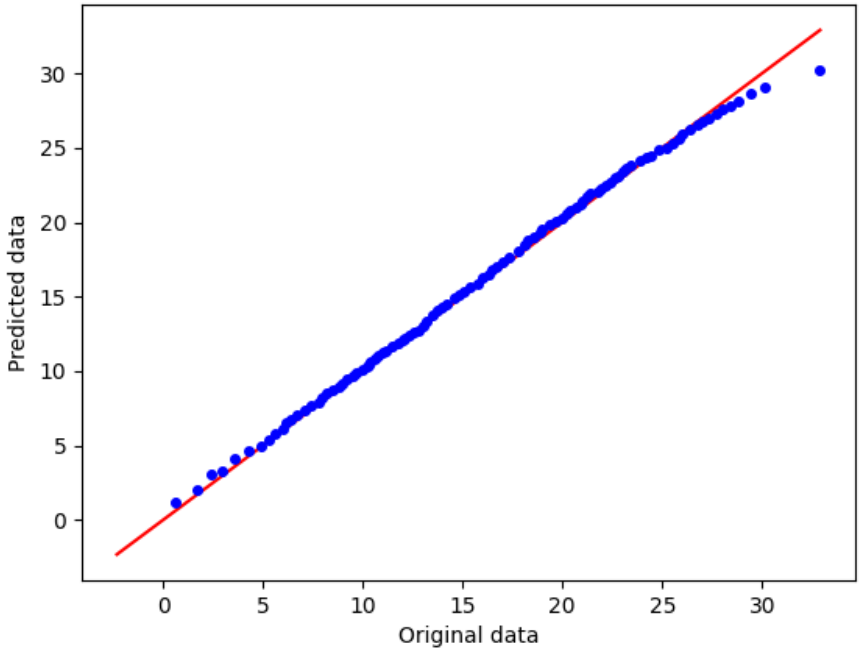

Figura 142: Gráfica Q-Q RNA con 2+1 días y 4 estaciones meteorológicas optimizada

RNA con 5+1 días y 4 estaciones meteorológicas

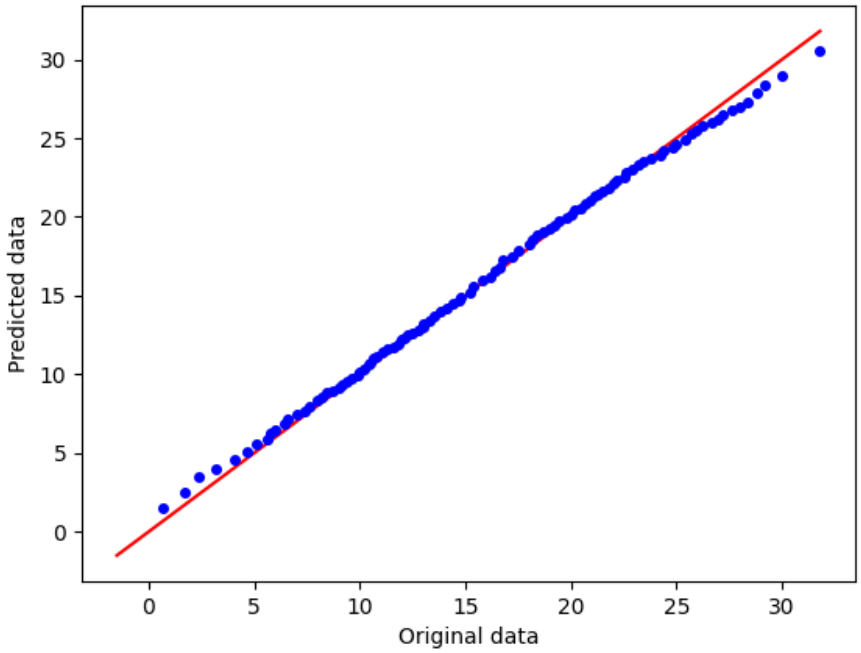

Figura 143: Gráfica Q-Q RNA con 5+1 días y 4 estaciones meteorológicas optimizada

RNA con 7+1 días y 4 estaciones meteorológicas

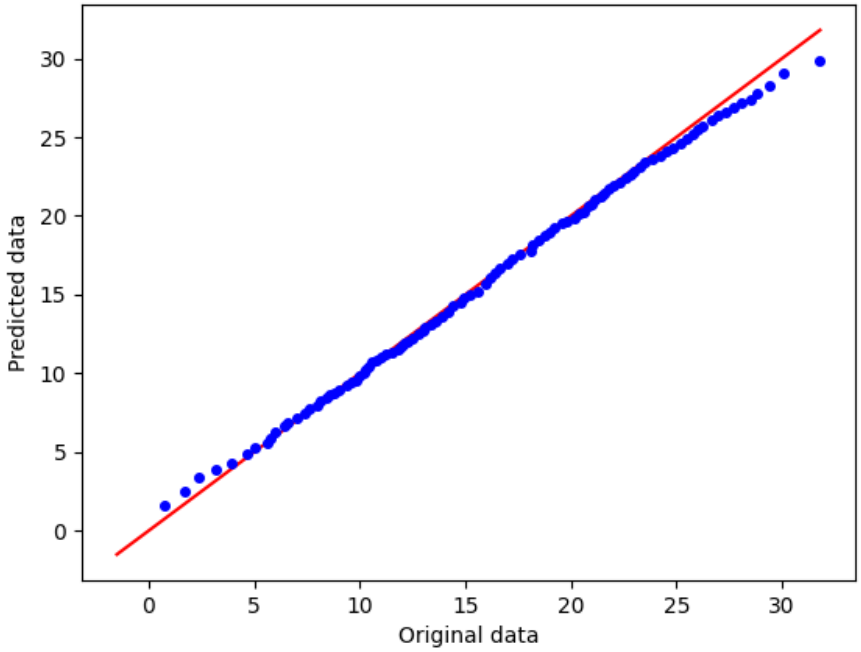

Figura 144: Gráfica Q-Q RNA con 7+1 días y 4 estaciones meteorológicas optimizada

RNA con 10+1 días y 4 estaciones meteorológicas

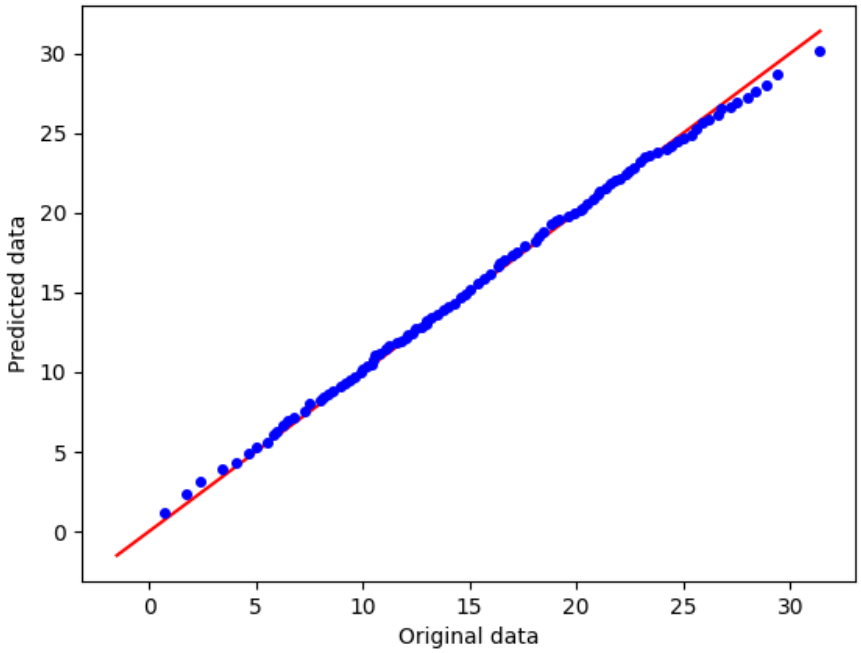

Figura 145: Gráfica Q-Q RNA con 10+1 días y 4 estaciones meteorológicas optimizada

RNA con 7+1 días, 4 estaciones meteorológicas y 300 de tamaño de lote

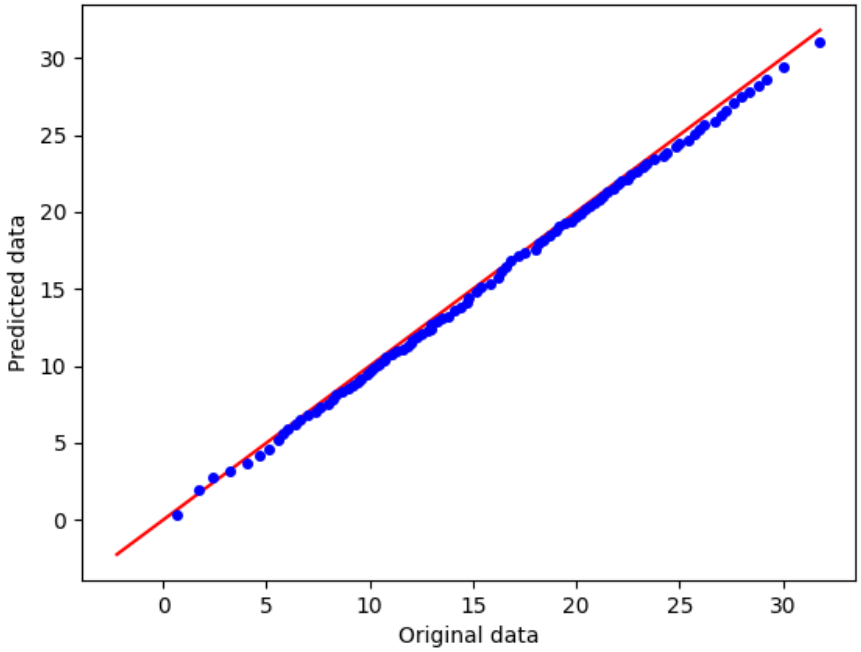

Figura 146: Gráfica Q-Q RNA con 7+1 días, 4 estaciones meteorológicas y 300 de tamaño de lote

Anexo J: MSE, RMSE, BIAS y Varianza

En esta sección se muestran los valores de: MSE, RMSE, BIAS y Varianza de cada RNA creada.

	RNA optimizada
RNA con 2+1 días y 1 estación meteorológica	Mean Square Error: 3.6070710943659656 Root Mean Square Error: 1.899229078959662 BIAS: 0.07774949692138122 Variance: 3.5293215974445844
RNA con 5+1 días y 1 estación meteorológica	Mean Square Error: 3.4754639919076467 Root Mean Square Error: 1.8642596363992991 BIAS: 0.17987014499242804 Variance: 3.2955938469152186
RNA con 7+1 días y 1 estación meteorológica	Mean Square Error: 3.4912533334875095 Root Mean Square Error: 1.8684895861330106 BIAS: -0.014358776299740583 Variance: 3.50561210978725
RNA con 10+1 días y 1 estación meteorológica	Mean Square Error: 3.484402836748852 Root Mean Square Error: 1.8666555217149339 BIAS: 0.2471749533116281 Variance: 3.237227883437224
RNA con 2+1 días y 4 estaciones meteorológicas	Mean Square Error: 3.7101261284967793 Root Mean Square Error: 1.926168769473947 BIAS: -0.17172173868858565 Variance: 3.881847867185365
RNA con 5+1 días y 4 estaciones meteorológicas	Mean Square Error: 3.699574732578513 Root Mean Square Error: 1.923427859988129 BIAS: 0.1655516769105212 Variance: 3.534023055667992
RNA con 7+1 días y 4 estaciones meteorológicas	Mean Square Error: 3.7026031234893155 Root Mean Square Error: 1.9242149369260482 BIAS: 0.060199556566757906 Variance: 3.6424035669225576
RNA con 10+1 días y 4 estaciones meteorológicas	Mean Square Error: 3.845369228394766 Root Mean Square Error: 1.9609613021155634 BIAS: -0.12178452075385238 Variance: 3.9671537491486184
RNA con 7+1 días, 4 estaciones meteorológicas y 300 de tamaño de lote	Mean Square Error: 4.436103695559715 Root Mean Square Error: 2.106205995518889 BIAS: 0.36916993751375315 Variance: 4.066933758045962

Tabla 7: MSE, RMSE, BIAS y Varianza de RNAs optimizadas