✳ FRP, UTN
DPTO. ING. ELECTROMECÁNICA

“Predicción de Series Temporales con Redes LSTM”

Nicolás Francisco Moguilner Reh

Informe Técnico de la Práctica Profesional Supervisada, presentado al Departamento de Ingeniería Electromecánica de la Facultad Regional de Paraná (FRP), Universidad Tecnológica Nacional (UTN), como parte de los requisitos para la carrera de Ingeniería Electromecánica. Las actividades asociadas a esta práctica se llevaron a cabo en el entorno de la FRP, bajo la supervisión del docente Hugo D. Pasinato.

Palabras clave: *Redes Neuronales Recurrentes (RNN), Memoria a Corto y Largo Plazo (LSTM), Predicción de Series Temporales, Ingeniería Electromecánica, Aprendizaje Automático (Machine Learning), Predicción de Demanda Eléctrica, Modelado de Fenómenos Dinámicos, Inteligencia Artificial, Análisis de Datos.*

Paraná, 24 de febrero de 2024

Resúmen

En este trabajo se exploró la aplicación de Redes Neuronales Recurrentes con Memoria a Corto y Largo Plazo (LSTM) para la predicción de series temporales y espaciales en ingeniería. Se analizó el funcionamiento de las LSTM, su formulación matemática y proceso de aprendizaje, así como su efectividad en la predicción de series temporales. Además, se llevó a cabo una prueba de concepto en el área eléctrica para predecir la demanda. Los resultados demostraron que las LSTM son una herramienta prometedora en la predicción de fenómenos complejos y pueden ofrecer soluciones precisas y flexibles para desafíos en ingeniería, destacando su potencial en la predicción de series temporales en diversos contextos.

Índice

Resúmen 2
1. Introducción 4
2. Fundamentos del tema 5
2.1. Perceptrones 5
2.2. Redes Neuronales y Backpropagation 5
2.3. Descenso por Gradiente 6
2.4. Función de Pérdida 6
2.5. Función de Activación 6
2.6. Redes Neuronales Recurrentes 7
3. Redes LSTM 8
3.1. Funcionamiento LSTM 9
3.1.1. Inicialización de Estados 9
3.1.2. Puertas de la LSTM 9
3.1.3. Actualización de la Celda de Memoria 11
3.1.4. Actualización del Estado Oculto y Salida 12
3.1.6. Repetición en cada Paso de Tiempo 12
3.2. Backpropagation en LSTMs. Formulación 13
3.3. Ventajas de usar LSTM para Predicción de Series Temporales 14
3.3.1. Importancia en Ingeniería 14
3.3.2. Aplicaciones en Ingeniería Electromecánica 18
4. Aplicación de LSTM para predecir consumo energético 20
4.1. Introducción 20
4.2. Detalles de la implementación 21
4.2.1. Módulos utilizados 21
4.2.2. Carga de datos 21
4.2.3 Algoritmo de Ventana Deslizante 23
4.2.4 Normalización de datos 24
4.2.5. Armado de sets de entrenamiento 24
4.2.6. Definición del modelo LSTM 24
4.2.7. Ajuste del modelo 26
4.3. Resultados 27
4.3.1. Coop. El Tala (Villa Urquiza) 27
4.3.2. Bovril 34
4.4. Conclusión 41
5. Bibliografía 42

1. Introducción

En el ámbito de la ingeniería, la búsqueda constante de soluciones innovadoras para problemas complejos ha impulsado la exploración de técnicas computacionales avanzadas. Destaca entre ellas el aprendizaje automático (Machine Learning), una disciplina de la inteligencia artificial que ha ganado prominencia en el procesamiento y análisis de datos. En el contexto del aprendizaje automático, las máquinas tienen la capacidad de analizar datos, aprender patrones a partir de experiencias pasadas y mejorar su rendimiento sin intervención humana directa, pudiendo realizar predicciones o tomar decisiones sin estar explícitamente programadas para tareas específicas.

Dentro del amplio espectro del aprendizaje automático, encontramos las redes neuronales, un componente clave en la aplicación de estas técnicas para el modelado de fenómenos dinámicos. Las redes neuronales son modelos computacionales inspirados en el funcionamiento del cerebro humano. Están compuestas por capas de nodos interconectados, llamados neuronas, que trabajan en conjunto para procesar y aprender patrones complejos a partir de datos de entrada. Cada conexión entre nodos tiene un peso que ajusta la influencia de una neurona sobre otra, permitiendo así que la red aprenda y adapte su comportamiento con el tiempo.

La aplicación del aprendizaje automático basado en redes neuronales busca emular la capacidad del cerebro para reconocer patrones y ejecutar tareas específicas. Esta aproximación ha adquirido un rol fundamental en diversas aplicaciones, aprovechando la habilidad de las redes neuronales para capturar relaciones no lineales y modelar fenómenos complejos que a menudo desafían métodos más convencionales. La versatilidad de estas redes abarca desde la visión por computadora hasta el procesamiento de lenguaje natural, destacando su utilidad en una variedad de contextos donde la interpretación de datos complejos resulta esencial[3].

Así, en el marco de las técnicas computacionales avanzadas, las redes neuronales se erigen como herramientas poderosas que, cuando se aplican de manera adecuada, pueden ofrecer soluciones flexibles y precisas para una variedad de desafíos en ingeniería. Este informe se centra específicamente en la aplicación de las Redes Neuronales Recurrentes con Memoria a Corto y Largo Plazo (LSTM), una variante de las redes neuronales, para abordar la predicción de series temporales y espaciales en el contexto de la ingeniería.

Este estudio se enfoca en analizar el funcionamiento de las redes LSTM, incluyendo su composición, formulación matemática y proceso de aprendizaje, así como su eficacia en la predicción de series temporales. Se comienza con una introducción conceptual a las redes neuronales en la siguiente sección, para luego profundizar en el tema en la sección 3. El núcleo de este trabajo se encuentra en la sección 4, donde se explora la aplicación de estas redes en la Ingeniería Electromecánica, específicamente en la predicción de la demanda eléctrica, seguido de las conclusiones obtenidas.

2. Fundamentos del tema

A continuación, se realiza una breve introducción a Redes Neuronales y otros conceptos relevantes.

2.1. Perceptrones

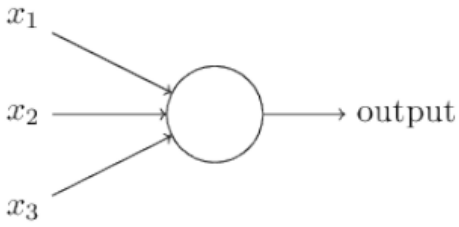

Fig. 1. Esquema de perceptrón[1].

El perceptrón, como unidad básica de procesamiento, es fundamental para entender las redes neuronales. Inspirado en la función de una neurona biológica, un perceptrón toma varias entradas, cada una multiplicada por un coeficiente correspondiente; este coeficiente se denomina peso. La suma ponderada se somete a una función de activación para producir la salida binaria del perceptrón.

De esta manera, la salida será 0 si la suma de sus entradas afectadas por su peso es menor o igual 0 o, será 1 si la suma es mayor a 0[1].

2.2. Redes Neuronales y Backpropagation

Las redes neuronales se componen de capas de perceptrones interconectados. Para entrenar una red neuronal, se utiliza el algoritmo de retropropagación (backpropagation). Este algoritmo ajusta los pesos de las conexiones para minimizar la diferencia entre las predicciones de la red y los resultados deseados.

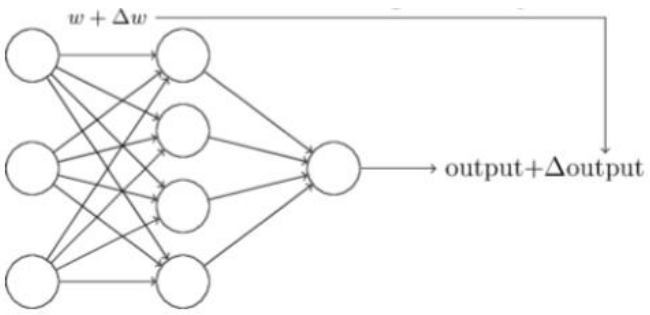

Fig. 2. Esquema de red neuronal. La primera capa corresponde a las entradas y la última a la respuesta de la red. La capa intermedia corresponde a las neuronas, denominada capa oculta. Una actualización de los pesos sinápticos dada por w + Δw se verá reflejada en una actualización de la salida[1].

El proceso de retropropagación implica calcular el gradiente de la función de pérdida con respecto a los pesos de la red. Luego, se utiliza el descenso por gradiente para ajustar los pesos en la dirección

que minimice la pérdida. Este proceso se repite iterativamente hasta que la red converge a un estado donde la pérdida es mínima.

2.3. Descenso por Gradiente

El descenso por gradiente es un método de optimización utilizado para encontrar el mínimo de una función. En el contexto de las redes neuronales, la función objetivo es la función de pérdida, que mide la discrepancia entre las predicciones de la red y los resultados reales.

La formulación matemática del descenso por gradiente está dada por

$$\theta = \theta - \alpha \nabla J(\theta) \tag{1}$$

, donde θ corresponde a los parámetros (pesos) de la red, α a la tasa de aprendizaje que controla el tamaño de los pasos de actualización, J(θ) es la función de pérdida y el término ∇J(θ) representa el gradiente de la función de pérdida con respecto a los parámetros θ.

2.4. Función de Pérdida

La función de pérdida evalúa qué tan bien la red está realizando predicciones en comparación con los resultados reales. Una función común es la de Error Cuadrático Medio o MSE, especialmente para problemas de regresión. Para un solo ejemplo de entrenamiento, el MSE se define como

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2}$$

, donde y_i es el valor real e $\hat{y}_i$ es la predicción de la red para el valor i.

2.5. Función de Activación

La función de activación determina la salida de una neurona o unidad en una red neuronal. Introduce no linealidades en el modelo, permitiendo a la red aprender patrones más complejos. La función de activación más comunmente utilizada es la función Sigmoide

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{3}$$

, pero existen otrás como la tangente hiperbólica (tanh) o la unidad lineal rectificada (ReLU).

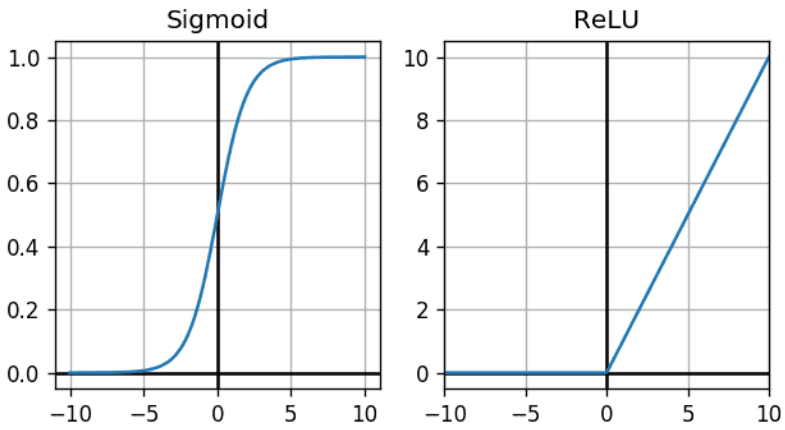

Fig. 3. Funciones de activación: Sigmoide (izquierda) y ReLU (derecha)[9].

2.6. Redes Neuronales Recurrentes

Las redes neuronales recurrentes (RNN) son un tipo de arquitectura de redes neuronales diseñadas para manejar datos secuenciales y modelar dependencias a lo largo del tiempo. A diferencia de las redes neuronales tradicionales, las RNN tienen conexiones que forman ciclos, permitiendo la retroalimentación de información de un paso de tiempo a otro. Esto las hace especialmente adecuadas para tareas que involucran secuencias, como el procesamiento del lenguaje natural, la traducción automática, la predicción de series temporales y otras aplicaciones en las que la información histórica es crucial.

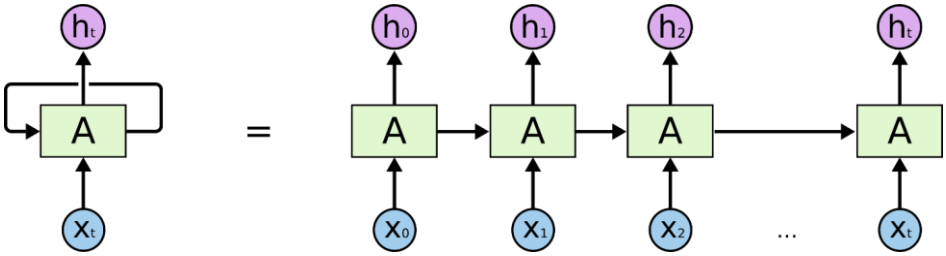

Fig. 4. Desdoblamiento de la recurrencia de una red neuronal[2].

Al igual que las redes neuronales tradicionales (feed-forward), las redes neuronales recurrentes utilizan datos de entrenamiento para aprender, pero se distinguen por su "memoria", ya que toman información de entradas anteriores para utilizarse en los datos de entrada y en los resultados.

Si bien las redes neuronales profundas tradicionales asumen que los datos de entrada y los resultados son independientes entre sí, los resultados de las redes neuronales recurrentes dependen de los elementos anteriores dentro de la secuencia. Aunque los eventos futuros también serían útiles para determinar los resultados de una secuencia dada, las redes neuronales recurrentes unidireccionales no pueden tener en cuenta estos eventos en sus predicciones[2].

3. Redes LSTM

Si bien las Redes Neuronales Recurrentes (RNN) son útiles para modelar datos secuenciales, se enfrentan a dificultades para capturar dependencias a largo plazo en las secuencias, ya que la información tiende a diluirse o perderse a medida que retrocede en el tiempo durante el entrenamiento. Como solución a este desafío, surgieron las Redes Neuronales Recurrentes con Memoria a Corto y Largo Plazo (LSTM), diseñadas con unidades de memoria especializadas que les permiten aprender y retener información relevante a lo largo de secuencias temporales prolongadas, lo que las hace más efectivas para tareas que requieren un procesamiento de datos secuenciales complejo y a largo plazo.

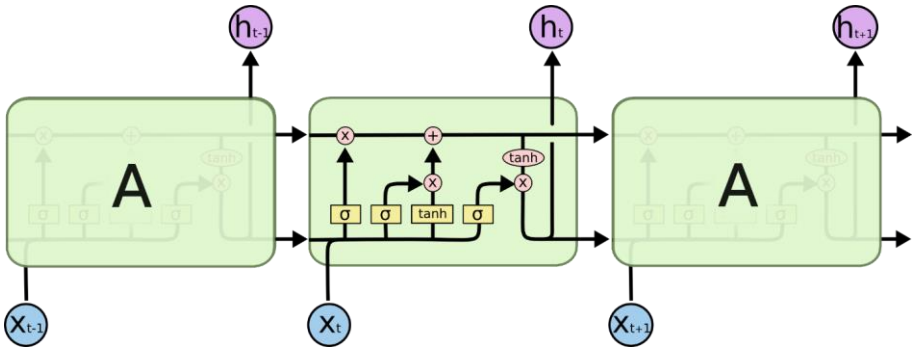

Fig. 5. Esquema de una LSTM y su configuración interna[2].

Estas unidades de memoria especiales son llamadas celdas de memoria, y están diseñadas para almacenar y recordar información a lo largo de largos períodos de tiempo.

Cada celda de memoria en una red neuronal recurrente está equipada con tres puertas que regulan el flujo de información hacia y desde la celda. Estas puertas son la puerta de olvido, que controla qué información debe ser olvidada o mantenida en la celda de memoria; la puerta de entrada, que decide qué nueva información debe agregarse a la celda de memoria; y la puerta de salida, que regula la información que se enviará como salida de la celda de memoria.

Las LSTMs ofrecen varias ventajas sobre las RNN tradicionales, especialmente al trabajar con series temporales. Esto incluye su capacidad para manejar dependencias a largo plazo, lo que las hace efectivas para capturar patrones en datos con relaciones temporales complejas y a largo plazo. Además, el diseño de las LSTMs con sus puertas de control previene la desaparición del gradiente, facilitando el entrenamiento de modelos en secuencias más largas y ayudando a preservar información relevante a lo largo del tiempo. También son adaptables a diferentes escalas temporales, lo que las hace versátiles para abordar tareas que involucran variaciones temporales a corto y largo plazo. Por último, debido a su capacidad para modelar dependencias temporales complejas, las LSTMs son ampliamente utilizadas en aplicaciones de series temporales como pronósticos meteorológicos, finanzas y procesamiento del lenguaje natural.

En resumen, las LSTMs han demostrado ser efectivas para modelar dependencias temporales a largo plazo, lo que las hace especialmente útiles en aplicaciones que implican el análisis de series temporales complejas y la captura de patrones a lo largo del tiempo.

3.1. Funcionamiento LSTM

3.1.1. Inicialización de Estados

En una red LSTM, los estados fundamentales son el estado oculto (h_t) y el estado de la celda (c_t). El estado oculto (h_t) en el tiempo t representa la información que la LSTM ha aprendido hasta ese momento y se utiliza como salida en ese paso de tiempo. Este estado influye en la siguiente salida y se actualiza para el siguiente paso de tiempo. Por otro lado, el estado de la celda (c_t) en el tiempo t es la memoria a largo plazo de la LSTM en ese momento. Contiene información relevante que la red ha decidido recordar de pasos de tiempo anteriores. Este estado se actualiza y modifica a lo largo del tiempo mediante las operaciones de las puertas y la actualización de la celda de memoria.

En la inicialización, generalmente en el primer paso de tiempo t=0, h_t y c_t se establecen en cero o en algún valor predeterminado. A medida que la red procesa la secuencia, estos estados se actualizan y almacenan información relevante para capturar dependencias a largo plazo en los datos secuenciales. La información en c_t puede persistir y ser utilizada para prever eventos en momentos futuros de la secuencia.

3.1.2. Puertas de la LSTM

La LSTM tiene la capacidad de eliminar o agregar información al estado de la celda, cuidadosamente regulada por estructuras llamadas puertas.

Las puertas son una forma de permitir opcionalmente el paso de información. Están compuestas por una capa de red neuronal sigmoidal y una operación de multiplicación punto a punto también conocida como producto Hadamard.

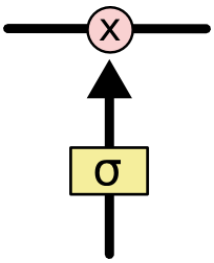

Fig. 6. Esquema de puerta en una LSTM[2].

La función sigmoide toma un valor real y lo "aplana" en un rango entre 0 y 1, por lo que su salida está siempre en dicho rango (0, 1). Se utiliza comúnmente en las puertas de una LSTM, como la puerta de olvido (ft), la puerta de entrada (it), y la puerta de salida (Ot), para controlar el flujo de información. La función sigmoide es particularmente útil para "decidir" qué información debe ser olvidada (en la puerta de olvido) o qué nueva información debe ser incluida (en la puerta de entrada).

Es decir, describe cuánto de cada componente debe permitirse pasar: un valor de cero significa "no permitir que pase nada", mientras que un valor de uno significa "permitir que pase todo".

Cada puerta tiene asociados pesos y funciones de activación.

En contraparte, La función tangente hiperbólica también aplana los valores, pero en un rango entre -1 y 1. Se utiliza en las LSTMs especialmente para calcular el candidato de la celda ($\tilde{C}t$) y actualizar el estado oculto (ht). La ventaja de tanh es que permite a la red aprender representaciones que incluyen valores negativos, lo que puede ser útil para capturar patrones más complejos en los datos.

Puerta de Olvido (Forget Gate)

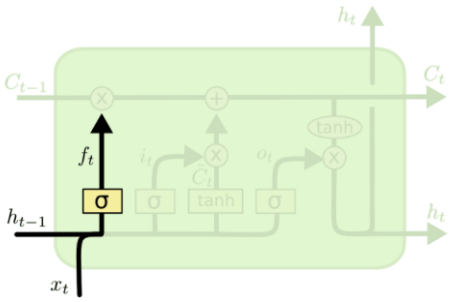

Fig. 7. Puerta de olvido o Forget gate[2].

Esta puerta determina qué información de la celda de memoria anterior (c_{t-1}) debe olvidarse.

Esta dada por la ecuación:

$$f_t = \sigma\left(W_f \cdot \left[h_{\{t-1\}}, x_t\right] + b_f\right) \tag{4}$$

donde Wf son los pesos, σ es la función sigmoidal, [ht−1,xt] es la concatenación del estado oculto anterior y la entrada actual, y bf es el sesgo.

Puerta de Entrada (Input Gate)

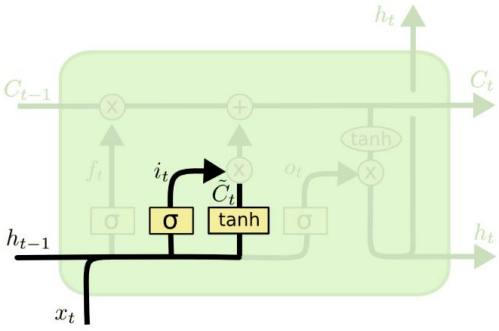

Fig. 8. Puerta de entrada o Input gate[2].

Esta puerta decide qué nueva información debe agregarse a la celda de memoria y está dada por la ecuación:

$$i_t = \sigma(W_i \cdot [h_{\{t-1\}}, x_t] + b_i) \quad (5)$$

Puerta de Salida (Output Gate)

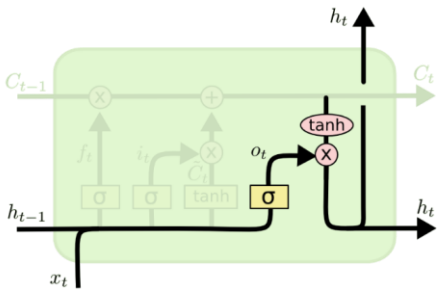

Fig. 9. Puerta de salida o Output gate[2].

La puerta de salida regula la información que se enviará como salida y actualiza el estado oculto.

$$o_t = \sigma(W_o \cdot [h_{\{t-1\}}, x_t] + b_o) \quad (6)$$

$$h_t = o_t * \tanh(C_t) \quad (7)$$

3.1.3. Actualización de la Celda de Memoria

La celda de memoria (ct) se actualiza utilizando la información de las puertas de olvido y entrada, junto con una versión candidata de la nueva información (c~t). La memoria candidata es la nueva versión de la memoria que pisa el estado C_{t-1} y lo actualiza a C_t.

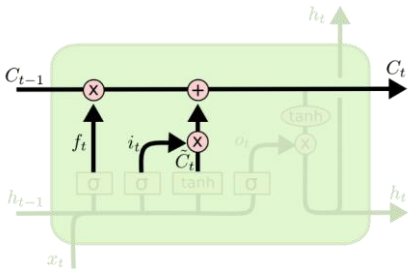

Fig. 10. Celda de memoria y puertas que permiten su actualización[2].

$$C_t = f_t \cdot C_{\{t-1\}} + i_t \cdot C\sim t \quad (8)$$

La memoria candidata (c~t) se calcula mediante:

$$C\sim t\ = \tanh\left(W_c \cdot \left[h_{\{t-1\}}, x_t\right] + \ b_c\right) \tag{9}$$

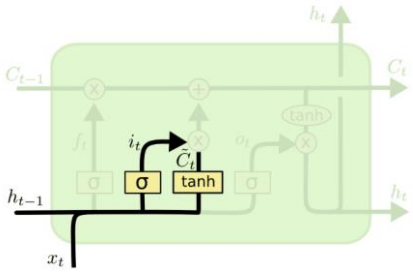

Fig. 11. Operación de selección de nueva memoria tentativa o actualización de la celda de memoria[2].

La función tangente hiperbólica tanh determina que tan sensible o relevante es el nuevo estado de memoria, al ser su salida entre -1 y 1.

3.1.4. Actualización del Estado Oculto y Salida

Según la ecuación (4), el estado oculto (ht) se actualiza en función de la celda de memoria actualizada y la puerta de salida.

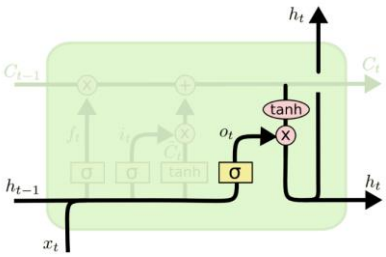

Fig. 12. Actualización del estado oculto h_t[2].

Este estado será la salida y_t de la unidad LSTM para ese momento t, de tal forma que $y(t) = output(h_t)$.

3.1.6. Repetición en cada Paso de Tiempo

Estos pasos se repiten en cada paso de tiempo de la secuencia, permitiendo que la LSTM capture dependencias a largo plazo y mantenga una memoria a largo plazo de la información relevante en la secuencia.

En resumen, las LSTMs utilizan puertas y celdas de memoria para controlar el flujo de información, recordar información a largo plazo y producir salidas en cada paso de tiempo de una secuencia. Esta

arquitectura las hace eficaces en la modelización de datos secuenciales con dependencias temporales a largo plazo.

3.2. Backpropagation en LSTMs. Formulación

La retropropagación a través del tiempo (BPTT) es un algoritmo utilizado para entrenar redes neuronales recurrentes (RNN). La BPTT en una LSTM implica propagar hacia atrás el error a lo largo de la secuencia de tiempo y ajustar los pesos de la red para minimizar ese error. A continuación, se detalla el proceso con su correspondiente formulación matemática:

En primer lugar, se realiza un pase hacia delante (forward pass) en célula LSTM. Las compuertas definidas en las ecuaciones 1, 2, 3 y 6 llevan a 5 y 4, donde se actualiza el estado de memoria interna y se obtiene la salida[5].

Luego, en la propagación hacia atrás (backward pass), siendo ΔT la diferencia de salida calculada por cualquier capa subsiguiente (es decir, el resto de la red) y Δh la diferencia de salida calculada por el siguiente paso temporal de la LSTM, se deberá hallar:

$$\delta h_t = \Delta_t + \Delta h_t \tag{10}$$

$$\delta C_t = \delta h_t * o_t * \left(1 - \tanh^2(C_t)\right) + \delta C_{t+1} * f_{t+1} \tag{11}$$

$$\delta C{\sim}t = \delta C_t * i_t * (1 - C{\sim}t^2) \tag{12}$$

$$\delta i_t = \delta C_t * \delta C{\sim}t * i_t * (1 - i_t) \tag{13}$$

$$\delta f_t = \delta C_t * \delta C_{t-1} * f_t * (1 - f_t) \tag{14}$$

$$\delta o_t = \delta h_t * \tanh(C_t) * o_t * (1 - o_t) \tag{15}$$

$$\delta x_t = W^T . \delta gates_t \tag{16}$$

$$\Delta h_{t-1} = U^T . \delta gates_t \tag{17}$$

Donde, para simplificar

$$gates_t = \begin{bmatrix} C{\sim}t \\ i_t \\ f_t \\ o_t \end{bmatrix}, W = \begin{bmatrix} W_{C{\sim}t} \\ W_i \\ W_f \\ W_o \end{bmatrix}, U = \begin{bmatrix} U_{C{\sim}t} \\ U_i \\ U_f \\ U_o \end{bmatrix}, b = \begin{bmatrix} b_{C{\sim}t} \\ b_i \\ b_f \\ b_o \end{bmatrix} \tag{18}$$

Siendo $U = \left[h_{\{t-1\}}, x_t\right]$.

Luego, las actualizaciones finales de los parámetros internos se calculan como:

$$\delta W = \sum_{t=0}^{T} gates_t \times X_t \tag{19}$$

$$\delta U = \sum_{t=0}^{T-1} gates_{t+1} \times h_t \tag{20}$$

$$\delta b = \sum_{t=0}^{T} gates_{t+1} \tag{21}$$

3.3. Ventajas de usar LSTM para Predicción de Series Temporales

Las Redes Neuronales Recurrentes con Memoria a Corto y Largo Plazo (LSTM) ofrecen una serie de ventajas significativas para la predicción de series temporales. En primer lugar, están diseñadas específicamente para capturar dependencias a largo plazo en datos secuenciales, lo cual es esencial dado que los patrones en las series temporales pueden extenderse a lo largo de períodos temporales prolongados. Además, las LSTM tienen la capacidad de modelar y adaptarse a patrones temporales complejos, incluidos aquellos en series temporales con irregularidades, cambios abruptos o patrones no lineales. Esto se logra mediante el uso de puertas, mecanismos internos que controlan qué información se debe olvidar, qué información se debe almacenar y cuánta información se debe utilizar en la salida, lo que facilita el modelado de relaciones complejas en las series temporales.

Otra ventaja importante es la adaptabilidad de las LSTM a diferentes escalas temporales, lo que les permite capturar tanto patrones a corto plazo como a largo plazo en los datos. Además, las LSTM son robustas ante el ruido en los datos y pueden aprender a filtrar información relevante incluso en presencia de variaciones no estructuradas en la serie temporal. Por último, la flexibilidad en la entrada y salida de las LSTM es destacable, ya que pueden manejar secuencias de longitud variable tanto en la entrada como en la salida, lo que permite la predicción de series temporales con diferentes longitudes. En resumen, las LSTM ofrecen una serie de características que las hacen altamente efectivas para la predicción de series temporales en una variedad de contextos.

3.3.1. Importancia en Ingeniería

A continuación, se listan una serie de ejemplos de aplicaciones relevantes de LSTMs en Ingeniería, que proporcionan una visión más completa y específica de cómo las LSTMs pueden aplicarse en una variedad de contextos, desde la gestión de energía hasta la agricultura de precisión, cubriendo áreas como la mecánica de fluidos, la ingeniería estructural y el control de procesos industriales.

Industria Energética

- Gestión de Energía: Previsión de demanda y optimización de producción en centrales eléctricas.
- Mantenimiento Predictivo de Equipos: Predicción de fallos en turbinas eólicas, generadores y transformadores.
- Planificación de Mantenimiento: Optimización de agendas de mantenimiento para minimizar tiempos de inactividad.

- Integración de Energías Renovables: Adaptación dinámica ante variabilidad en la generación de energía solar y eólica.
- Detección de Fraudes Energéticos: Identificación de patrones irregulares en el consumo para prevenir fraudes.
- Detección de Anomalías en la Red: Identificación de patrones temporales asociados a problemas en la red eléctrica.
- Optimización del Consumo: Estimación de patrones de consumo para mejorar la eficiencia energética.
- Estabilidad de la Red: Modelado temporal para prever y mitigar problemas de estabilidad en la red eléctrica.

Manufactura

- Pronóstico de Fallas: Predicción de desgaste en maquinaria pesada y sistemas de producción.
- Optimización de Procesos de Producción: Ajuste en tiempo real de líneas de producción para maximizar eficiencia.
- Gestión de Calidad: Monitoreo de variables para mantener estándares de calidad en productos manufacturados.
- Eficiencia Energética: Optimización del consumo de energía en plantas de manufactura.
- Detección de Defectos: Identificación temprana de defectos en productos mediante análisis temporal.

Mecánica de Fluidos

- Diseño de Vehículos Aerodinámicos: Optimización de perfiles aerodinámicos para vehículos terrestres y aéreos.
- Simulación de Impacto Ambiental: Evaluación de efectos ambientales en proyectos que involucran fluidos.
- Mejora en Eficiencia de Turbinas: Ajuste dinámico de álabes para maximizar eficiencia en turbinas hidroeléctricas.
- Predicción de Ondas y Mareas: Modelado de patrones de mareas y su impacto en estructuras marítimas.
- Modelado de la Turbulencia: Análisis temporal de patrones de flujo turbulento para diseño eficiente.

Ingeniería Estructural

- Diseño de Materiales Inteligentes: Creación de materiales adaptables a cambios temporales en carga y condiciones ambientales.
- Pronóstico de Degradación de Infraestructuras: Anticipación de deterioro en puentes, carreteras y edificios.
- Optimización de Mantenimiento de Infraestructuras: Programación eficiente de mantenimiento basada en predicciones temporales.
- Simulación de Cargas Dinámicas: Modelado de efectos temporales de cargas dinámicas en estructuras.

- Análisis de Vibraciones: Evaluación de patrones temporales de vibraciones para identificar posibles problemas estructurales.

Control de Procesos Industriales

- Automatización de Procesos Químicos: Ajuste dinámico de parámetros en tiempo real para mantener calidad y seguridad.
- Optimización de Producción en Tiempo Real: Mejora continua de procesos industriales en función de datos temporales.
- Gestión de Calidad: Monitoreo continuo de variables para asegurar estándares de calidad.
- Reducción de Desperdicios: Identificación de patrones temporales para minimizar desperdicios en la producción.
- Adaptación a Cambios en la Demanda: Ajuste dinámico de la producción según fluctuaciones temporales en la demanda.

Detección de Anomalías en Equipos

- Seguridad en Equipos: Identificación de patrones previos a situaciones de riesgo, como sobrecalentamiento o desgaste.
- Mantenimiento Predictivo: Anticipación de fallos en maquinaria industrial para programar mantenimiento.
- Eficiencia en Operación: Monitoreo continuo para mejorar la eficiencia operativa y evitar tiempos de inactividad.
- Gestión de Activos: Evaluación de patrones temporales para optimizar la vida útil de equipos.
- Detección de Fallas Eléctricas: Identificación de patrones temporales asociados a problemas eléctricos en equipos.

Optimización de Cadenas de Suministro en Transporte

- Gestión de Inventarios: Predicción de la demanda para optimizar niveles de inventario.
- Ruteo Eficiente: Planificación dinámica de rutas para minimizar costos y tiempos de entrega.
- Adaptación a Variaciones Temporales: Ajuste dinámico de estrategias logísticas en función de patrones temporales.
- Prevención de Cuellos de Botella: Identificación temprana de posibles congestiones en la cadena de suministro.
- Eficiencia en Distribución: Optimización de procesos de distribución considerando variaciones temporales en la demanda.

Agricultura de Precisión

- Monitoreo de Nutrientes del Suelo: Análisis temporal para optimizar la distribución de nutrientes en los cultivos.
- Gestión del Riego: Predicción de necesidades de riego basada en patrones temporales climáticos y de suelo.

- Pronóstico de Rendimientos: Estimación de cosechas futuras para planificación de la producción agrícola.
- Adaptación a Condiciones Climáticas: Ajuste dinámico de prácticas agrícolas según variaciones temporales en el clima.
- Detección de Plagas: Identificación de patrones temporales asociados a la presencia de plagas para medidas preventivas.

Pronóstico Financiero

- Predicción de Tasas de Interés: Estimación de cambios en tasas de interés para planificación financiera.
- Análisis de Riesgos: Evaluación de riesgos financieros mediante predicciones de mercados.
- Predicción de Inversiones: Asistencia en decisiones de inversión basadas en tendencias temporales.
- Modelado de Portafolios: Optimización de carteras de inversión considerando fluctuaciones temporales.
- Predicción de Fluctuaciones Cambiarias: Estimación de cambios en tasas de cambio para gestión de divisas.

Meteorología

- Predicciones Climáticas Específicas: Pronósticos detallados para actividades específicas como eventos deportivos.
- Gestión de Recursos Hídricos: Predicción de lluvias para planificación de almacenamiento de agua.
- Seguridad en Desplazamientos: Información precisa sobre condiciones climáticas para seguridad vial.
- Alertas Tempranas de Desastres: Identificación anticipada de patrones asociados a desastres naturales.
- Simulación de Cambio Climático: Modelado a largo plazo para entender y abordar el cambio climático.

Salud

- Detección de Patrones en Enfermedades: Identificación de patrones temporales en registros médicos para detección temprana.
- Pronóstico de Evolución de Enfermedades: Estimación de la progresión de enfermedades crónicas.
- Adaptación de Tratamientos: Ajuste dinámico de tratamientos basado en respuestas temporales del paciente.
- Gestión de Recursos Hospitalarios: Predicción de demanda de servicios de salud para una asignación eficiente de recursos.
- Prevención de Epidemias: Monitoreo de patrones de propagación para prevención y control de epidemias.

3.3.2. Aplicaciones en Ingeniería Electromecánica

De manera más específica, en el ámbito de la Ingeniería Electromecánica, las redes LSTM ofrecen un amplio espectro de aplicaciones prometedoras.

Gestión Energética y Máquinas Eléctricas

- Pronóstico de Consumo Eléctrico: Utilización de LSTMs para prever la demanda de energía y optimizar la generación.
- Mantenimiento Predictivo en Generadores: Predicción de fallas en generadores eléctricos para programar intervenciones de mantenimiento.
- Eficiencia en Motores Eléctricos: Ajuste dinámico de parámetros para optimizar el rendimiento de motores eléctricos.

Mecánica y Elementos de Máquina

- Pronóstico de Desgaste en Engranajes: Predicción de la vida útil y desgaste en sistemas de engranajes.
- Mantenimiento Predictivo en Sistemas Mecánicos: Identificación de patrones que indican posibles fallos en componentes mecánicos.
- Optimización de Elementos de Máquina: Ajuste dinámico de parámetros para maximizar la eficiencia y durabilidad.

Termodinámica y Máquinas Térmicas

- Eficiencia en Ciclos Termodinámicos: Ajuste de variables para optimizar la eficiencia en ciclos de máquinas térmicas.
- Pronóstico de Rendimiento en Calderas: Utilización de LSTMs para prever cambios en el rendimiento de calderas y optimizar la combustión.
- Mantenimiento Predictivo en Turbinas de Vapor: Identificación de patrones asociados a posibles fallas en turbinas de vapor.

Mecánica de Fluidos y Máquinas Hidrodinámicas

- Optimización de Perfiles en Hélices: Ajuste dinámico de perfiles en hélices para maximizar la eficiencia en sistemas propulsores.
- Pronóstico de Rendimiento en Bombas: Utilización de LSTMs para prever cambios en el rendimiento de bombas hidráulicas.
- Mantenimiento Predictivo en Sistemas Hidrodinámicos: Identificación de patrones que indican posibles problemas en sistemas hidrodinámicos.

Instalaciones y Máquinas Frigoríficas

- Eficiencia en Ciclos de Refrigeración: Ajuste dinámico de variables para optimizar la eficiencia en ciclos de máquinas frigoríficas.
- Pronóstico de Consumo en Sistemas de Climatización: Utilización de LSTMs para prever la demanda de energía en sistemas de climatización.
- Mantenimiento Predictivo en Compresores: Identificación de patrones asociados a posibles fallas en compresores de sistemas de refrigeración.

Procesos Industriales y Proyectos de Inversión

- Optimización de Procesos: Utilización de LSTMs para ajustar dinámicamente los parámetros en procesos industriales, mejorando eficiencia y reduciendo costos operativos.
- Evaluación de Riesgos en Proyectos: Análisis temporal para evaluar riesgos y rendimientos en proyectos de inversión a largo plazo.
- Análisis de Rentabilidad en Inversiones: Utilización de LSTMs para prever rendimientos financieros y evaluar la viabilidad de proyectos de inversión.

Manejo de Materiales

- Gestión de Inventario: Pronóstico de demanda para optimizar niveles de inventario y asegurar disponibilidad de materiales.
- Mantenimiento Predictivo en Equipos de Manejo de Materiales: Identificación de patrones que indican posibles fallas en equipos de manejo de materiales.
- Rastreo de Productos en Tiempo Real: Utilización de LSTMs para seguir y predecir el flujo de materiales a través de la cadena de suministro.

4. Aplicación de LSTM para predecir consumo energético

4.1. Introducción

Antes de la proliferación de las redes neuronales y las técnicas de aprendizaje profundo, las estrategias más comunes para la predicción de series temporales se basaban en enfoques estadísticos tradicionales. Entre estas estrategias se encontraban los modelos ARIMA (Autoregressive Integrated Moving Average), que utilizaban la autocorrelación y la autocorrelación parcial de la serie temporal para modelar tendencias, estacionalidad y componentes residuales. Los modelos de regresión lineal y no lineal eran ampliamente empleados para predecir valores futuros en función de variables predictoras. Estas técnicas estadísticas tradicionales han sido fundamentales en el análisis de series temporales durante décadas y siguen siendo relevantes en muchos contextos. Sin embargo, desde el surgimiento de la Inteligencia Artificial (IA), el uso de redes neuronales, particularmente las redes neuronales recurrentes como la LSTM, han demostrado una superioridad y mayor simplicidad en muchos casos. Esto ha proporcionado nuevas perspectivas y capacidades para abordar problemas de predicción de series temporales de manera más efectiva y eficiente, tanto en ingeniería y todas sus ramas como en otras disciplinas[11].

Este estudio se basa en una sólida prueba de concepto que emplea datos proporcionados por ENERSA, los cuales consisten en mediciones mensuales de potencia y corriente desde el año 2004 hasta la fecha. Estos datos provienen de varios municipios y zonas de la provincia de Entre Ríos, ofreciendo una representación significativa de la variabilidad en el consumo de energía en la región.

El trabajo se centra en la implementación de redes LSTM para la predicción mensual de consumo de potencia. Utilizando las tablas de Excel proporcionadas por ENERSA como conjunto de datos de entrenamiento, se logra capacitar el modelo para capturar patrones complejos y dependencias temporales presentes en las mediciones mensuales.

Es crucial destacar que los resultados obtenidos no arrojan conclusiones definitivas ni son directamente aplicables a la realidad sin una revisión exhaustiva. Aunque se demuestra la eficacia relativa de los modelos LSTM en la predicción, se necesita un análisis más profundo antes de su implementación práctica para que este enfoque sea aplicable e incluso rentable. Esto incluye la recopilación de un volumen aún mayor de datos, la participación de un equipo dedicado exclusivamente al estudio del comportamiento de diversos modelos y la evaluación continua de su precisión. Además, cualquier modelo desarrollado debería someterse a auditorías rigurosas para garantizar su confiabilidad y robustez.

Es importante subrayar que, incluso con un análisis detallado, existe un margen de error inherente a cualquier modelo predictivo. Las predicciones están condicionadas por la calidad y representatividad de los datos utilizados en el entrenamiento del modelo. Además, eventos inesperados y situaciones futuras impredecibles, como la pandemia de coronavirus entre los años 2020 y 2022, ilustran la dificultad de anticipar todos los posibles escenarios en la realidad.

En resumen, aunque la prueba de concepto proporciona una visión alentadora de las posibilidades de los modelos LSTM en la predicción de la demanda, se subraya la necesidad de proceder con cautela y reconocer las limitaciones inherentes. Un enfoque meticuloso y colaborativo sería esencial para

desarrollar e implementar con éxito un sistema predictivo que pueda ser confiable en entornos dinámicos y cambiantes como el sector de la ingeniería electromecánica.

4.2. Detalles de la implementación

El programa se centra en la implementación de un modelo de redes neuronales LSTM para predecir valores futuros en una serie temporal de datos relacionados con la potencia activa. El proceso implica una serie de pasos detallados que se describen a continuación, destacando los aspectos más relevantes.

4.2.1. Módulos utilizados

Inicialmente, se importan los módulos o librerías necesarias, como NumPy, TensorFlow, pandas, Matplotlib y otras utilidades específicas para el procesamiento de datos y la construcción de modelos de redes neuronales, para establecer la base sobre la cual se desarrollará el proyecto. Estas herramientas proporcionan las capacidades fundamentales para manipular y analizar los datos, así como para construir y entrenar modelos de aprendizaje automático.

En la Tabla 1 se describen brevemente cada una de las librerías utilizadas a lo largo del programa.

Librería	**Función**
NumPy	Proporciona funciones para la manipulación de matrices y operaciones numéricas eficientes.
TensorFlow	Un framework de aprendizaje automático de código abierto desarrollado por Google para construir modelos de ML.
Pandas	Ofrece estructuras de datos flexibles y herramientas para el análisis de datos.
Matplotlib	Permite crear visualizaciones estáticas, gráficos y diagramas a partir de datos.
scikit-learn	Proporciona herramientas simples y eficientes para la minería y el análisis de datos.
Seaborn	Construye visualizaciones atractivas y informativas de datos estadísticos.
os	Proporciona funciones para interactuar con el sistema operativo.
datetime	Permite manipular fechas y horas en Python.

Tabla 1. Bibliotecas utilizadas en el programa.

4.2.2. Carga de datos

Una vez que las bibliotecas están cargadas, el siguiente paso crítico es la preparación de los datos. Se carga un conjunto de datos desde un archivo Excel que contiene registros de la potencia activa en diversas ubicaciones a lo largo del tiempo. Esta etapa es crucial ya que garantiza que los datos estén en un formato adecuado para su análisis posterior, lo que implica realizar operaciones de limpieza y

preprocesamiento, como la eliminación de entradas redundantes o inconsistentes, el manejo de valores faltantes mediante interpolación y la organización de los datos en una estructura coherente y accesible.

Al finalizar este proceso se obtiene un DataFrame (Fig. 1 y 2), es decir, una tabla en memoria donde cada fila corresponde a una muestra mensual, comenzando el día 1ero de enero de 2004 y finalizando el 1ero de agosto de 2023. La primera columna corresponde a la fecha de la muestra y cada una de las siguientes columnas subsiguientes corresponden a una ubicación o localidad. El valor en cada celda para cada una de estas columnas corresponde a la potencia activa.

Index	Date	4 Bocas (Escriña) pot	Alcaraz pot	Barrio Nautico (Villa Paranacito) pot	aylina pc	erduc pc	rnardis p
0	2004-01-01 00:00:00	40.143	937.38	290.793	1059.27	76.431	59.866
1	2004-02-01 00:00:00	38.658	953.62	284.688	1024.76	484.979	54.935
2	2004-03-01 00:00:00	38.021	397.061	306.241	636.998	517.539	51.581
3	2004-04-01 00:00:00	36.534	410.886	58.954	350.027	53.909	52.696
4	2004-05-01 00:00:00	35.045	401.233	57.326	198.444	442.014	58.301
5	2004-06-01 00:00:00	36.214	429.521	57.326	136.616	572.86	60.096
6	2004-07-01 00:00:00	40.145	891.35	266.658	142.447	576.121	146.992
7	2004-08-01 00:00:00	39.825	438.107	284.888	138.592	569.615	64.577
8	2004-09-01 00:00:00	36.744	447.098	275.591	144.444	589.793	65.933
9	2004-10-01 00:00:00	38.337	889.96	270.452	230.31	79.893	118.181
10	2004-11-01 00:00:00	39.932	463.808	287.697	558.268	574.182	51.339
11	2004-12-01 00:00:00	43.327	495.228	279.744	1188.72	689.395	58.06
12	2005-01-01 00:00:00	48.214	486.21	304.034	1230.43	25.916	74.665
13	2005-02-01 00:00:00	40.25	467.322	322.939	1143.86	86.154	66.392

Fig. 13. DataFrame que contiene los valores de potencia activa para cada localidad en cada mes.

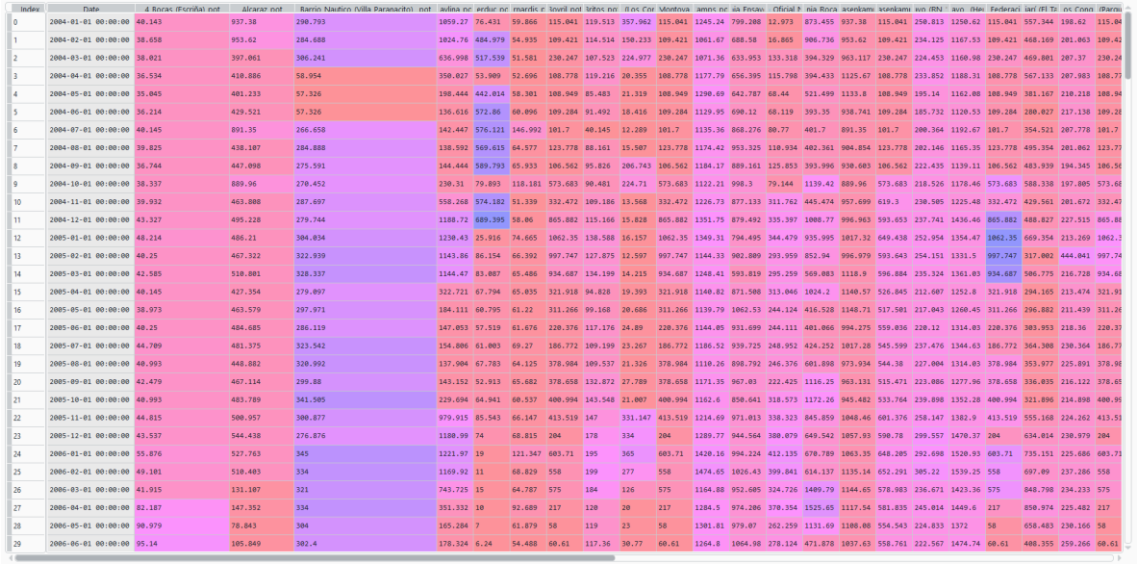

Fig. 14. Visión extendida del DataFrame generado para dimensionar la cantidad de registros a ser utilizados.

4.2.3 Algoritmo de Ventana Deslizante

Posteriormente, se emplea un algoritmo de "ventana deslizante" (Fig. 15) para la construcción de los datos de entrenamiento. Esta función se encarga de organizar los datos temporales de manera que puedan ser utilizados para el entrenamiento del modelo LSTM. Cada ventana de datos comprende un número específico de pasos temporales anteriores (look_back), y se crea un conjunto de secuencias de entrada (dataX) y sus correspondientes etiquetas de salida (dataY). La idea principal detrás de la ventana deslizante es transformar la serie temporal en un formato que el modelo pueda entender, dividiéndola en secuencias más pequeñas y etiquetando cada secuencia con el siguiente valor en la serie.

Este paso es crucial, ya que afecta directamente el proceso de entrenamiento. La selección de los pasos temporales, es decir, cuántos pasos hacia atrás se deben considerar para predecir un valor futuro, generalmente se determina mediante un proceso de prueba y error, comenzando con valores mínimos. Un valor de "look back" demasiado grande puede conducir a un aprendizaje deficiente.

```python
def create_dataset(dataset, look_back=1):
    dataX, dataY = [], []
    for i in range(len(dataset)-look_back-1):
        a = dataset[i:(i+look_back), 0]
        dataX.append(a)
        dataY.append(dataset[i + look_back, 0])
    return np.array(dataX), np.array(dataY)
```

Fig. 15. Fragmento de código; función que crea el set de datos X e Y a ser utilizados en el entrenamiento del modelo.

La función create_dataset definida en la Fig. 15, recibe como argumento el conjunto de datos, que puede ser un array unidimensional o bidimensional de numpy, y el parámetro look_back, que indica el número de pasos temporales anteriores que se utilizarán para predecir el siguiente valor, la función inicia su proceso de creación de conjuntos de datos de entrenamiento.

En su funcionamiento, la función itera a través de los datos desde el inicio hasta la longitud final de arreglo menos el número de look-back. Esta iteración asegura que se puedan formar ventanas de datos de la longitud adecuada, evitando índices fuera de rango.

En cada iteración, se crea una ventana de datos de longitud igual a look_back, comenzando desde el índice actual i hasta i + look_back. Esta ventana se añade a la lista dataX, que contendrá las secuencias de entrada para el modelo LSTM.

Posteriormente, se obtiene la etiqueta de salida correspondiente a la ventana de datos actual. Esta etiqueta se encuentra en el índice i + look_back y se añade a la lista dataY.

Una vez completada la iteración a través de los datos, las listas dataX y dataY se convierten en arrays de numpy utilizando np.array(). Estos arrays se devuelven como resultado de la función, donde dataX contiene las secuencias de entrada y dataY contiene las etiquetas de salida correspondientes.

En resumen, la función create_dataset es esencial para la preparación de los datos temporales en el contexto de la modelización LSTM, ya que estructura los datos de manera adecuada para el entrenamiento del modelo, permitiendo que capture y aprenda patrones y relaciones temporales relevantes en la serie de datos.

4.2.4 Normalización de datos

En un paso siguiente, se estandarizan o normalizan los datos de entrenamiento antes de alimentarlos al modelo. La normalización es un paso común en el preprocesamiento de datos para redes neuronales y tiene varios propósitos como ayudar a estabilizar el proceso de entrenamiento permitiendo una convergencia más rápida, mejorar el desempeño del modelo y mitigar problemas numéricos.

Como se observa en la Fig. 16, al normalizar los datos utilizando StandardScaler, se centran y escalan los datos para que tengan una media de cero y una desviación estándar de uno.

```
dataset = dataframe.values
dataset = dataset.astype('float32')

scaler = StandardScaler()
dataset = scaler.fit_transform(dataset)
```

Fig. 16. Fragmento de código; se estandarizan los datos que se utilizaran en el entrenamiento.

4.2.5. Armado de sets de entrenamiento

Una vez los datos se encuentran normalizados se arman los sets de entrenamiento correspondientes para X e Y (Fig. 17), donde “trainX” contiene las características de entrada y “trainY” contiene las etiquetas o valores objetivo correspondientes. Estos sets se utilizan juntos durante el proceso de entrenamiento para ajustar los parámetros del modelo.

```
train_size = int(len(dataset))
train = dataset[0:train_size,:]

trainX, trainY = create_dataset(train, look_back)
trainX = np.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))
```

Fig. 17. Fragmento de código; se determina el tamaño de la muestra de entrenamiento y se arman los sets de entrenamiento correspondientes a X e Y.

4.2.6. Definición del modelo LSTM

La arquitectura del modelo LSTM se define utilizando la biblioteca TensorFlow, listada en la Tabla 1. En esta implementación se utiliza un modelo secuencial que consta de dos capas LSTM apiladas y una capa densa o completamente conectada a la salida.

Capa	Neuronas	Función
Bidirectional	32	Capa LSTM bidireccional que procesa la secuencia de entrada tanto en orden directo como en orden inverso.
Bidirectional	64	Capa LSTM bidireccional que procesa la secuencia de entrada tanto en orden directo como en orden inverso.
Dropout	NA	Capa de dropout que apaga aleatoriamente un porcentaje de neuronas durante el entrenamiento para prevenir el sobreajuste.
Dense	1	Capa densa o completamente conectada que produce la salida final del modelo.

Tabla 2. Arquitectura del modelo LSTM utilizado.

Como se detalla en la Tabla 2, La primera capa LSTM tiene 64 neuronas, lo que significa que tiene 64 unidades de memoria, y es bidireccional. El parámetro "Bidireccional" permite que la capa procese la secuencia de entrada tanto en orden directo como en orden inverso, combinando la información de ambas direcciones. La segunda capa tiene la mitad de las neuronas, con un total de 32 unidades de memoria.

La última capa consta de una sola neurona y se utiliza para predecir un valor continuo (regresión); no se aplica una función de activación específica por lo que su sailda es lineal.

Entre la segunda capa LSTM y la capa completamente conectada se añade una capa de "Dropout", que ayuda a prevenir el sobreajuste (overfitting) al apagar aleatoriamente un porcentaje de neuronas durante el entrenamiento.

```
# create and fit the LSTM network
model = Sequential()
model.add(Bidirectional(LSTM(64, input_shape=(1, look_back), return_sequences=True)))
model.add(Bidirectional(LSTM(32, input_shape=(1, look_back),return_sequences=False)))
model.add(Dropout(0.2))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
```

Fig. 18. Fragmento de código; definición del modelo LSTM utilizado.

Una vez definido el modelo, este debe compilarse. Compilar el modelo en el contexto del aprendizaje profundo, se refiere al proceso de configurar las especificaciones necesarias para entrenar la red neuronal. Cuando se compila, se está definiendo cómo se realizará el proceso de entrenamiento.

En el caso del modelo utilizado para este programa, se compila utilizando el optimizador Adam y como función de pérdida se utiliza el error cuadrático medio o MSE (Mean Squared Error) (Fig. 18).

El optimizador "Adam" es un algoritmo de optimización ampliamente utilizado en el entrenamiento de modelos de aprendizaje profundo. Se adapta de forma dinámica durante el entrenamiento y ajusta las tasas de aprendizaje (learning rate) de cada parámetro por separado, lo que lo hace eficiente y efectivo en muchos escenarios. De esta forma, determina cómo se ajustan los pesos del modelo en función de la función de pérdida.

Por otra parte, el error cuadrático medio es la función de pérdida o costo utilizada en este caso. La misma es una medida que cuantifica cuán bien está realizando el modelo en sus predicciones en comparación con los valores reales. En este caso, se utiliza el MSE ya que es apropiado para problemas

de regresión. Mide el promedio de los cuadrados de las diferencias entre las predicciones y los valores reales.

4.2.7. Ajuste del modelo

Finalmente, se entrena el modelo considerando un porcentaje de datos de entrenamiento-validación de 80%-20% y se grafican las curvas de ajuste para analizar su convergencia.

```
history = model.fit(trainX, trainY, epochs=100, batch_size=32, verbose=2, validation_split=0.2)

plt.plot(history.history['loss'], label='Training loss')
plt.plot(history.history['val_loss'], label='Validation loss')
```

Fig. 19. Fragmento de código; entrenamiento del modelo y graficado.

Tomar una división del 80-20 en el conjunto de datos de entrenamiento y validación es una práctica estándar en el aprendizaje automático y el aprendizaje profundo. Esta división implica reservar el 80% de los datos para entrenamiento y el 20% restante para validación. Permite evaluar el rendimiento del modelo en datos no vistos durante el entrenamiento, prevenir el sobreajuste al monitorear el rendimiento en el conjunto de validación y ajustar los hiperparámetros del modelo.

Como se observa en la Fig. 19, se utiliza la función “fit” para ejecutar el entrenamiento del modelo o ajuste. Otros parámetros relevantes utilizados son el número de epócas o epochs y el tamaño de batch. En la Tabla 3 se describen brevemente.

Parámetro	Descripción	Valor Utilizado	Análisis
trainX	Datos de entrada utilizados para entrenar el modelo.	Datos específicos de entrenamiento	N/A
trainY	Etiquetas o valores objetivo correspondientes a los datos de entrada.	Etiquetas específicas de entrenamiento	N/A
epochs	Número de veces que el modelo se entrenará en el conjunto de datos completo.	100	Un valor común para permitir que el modelo se entrene a través del conjunto de datos múltiples veces y así mejorar el aprendizaje.
Batch_size	Número de muestras utilizadas en cada iteración de entrenamiento.	32	Un valor típico que equilibra eficiencia y estabilidad del entrenamiento.
Validation_split	Proporción del conjunto de entrenamiento que se utilizará como conjunto de validación.	0.2	Una división común para reservar una parte significativa de los datos de entrenamiento para la validación, ayudando a evaluar la generalización del modelo.

Verbose	Nivel de detalle de la información impresa durante el entrenamiento.	2	Un nivel moderado de verbosidad que muestra una barra de progreso durante el entrenamiento.

Tabla 3. Parámetros utilizados para el ajuste del modelo, una vez definida su arquitectura.

4.3. Resultados

Una vez ajustado el modelo, se grafican predicciones tomando como punto de partida la última muestra disponible de agosto de 2023 y proyectando a uno, dos, tres y diez años. A modo de validación visual del modelo también se predicen series temporales tomando puntos de partida anteriores, contrastándolas contra los datos históricos conocidos para estudiar si coinciden o si se presentan desviaciones.

Si bien se entrenó un modelo por cada una de las localidades disponibles en los datos proporcionados por ENERSA, para este informe se seleccionaron solo dos casos representativos. Las localidades analizadas son Coop. El Tala (Villa Urquiza) y Bóvril. A continuación, se detallan los sets de gráficas para cada una de ellas.

4.3.1. Coop. El Tala (Villa Urquiza)

En la Fig. 20 se puede observar la demanda de potencia activa para la localidad de Coop. El Tala (Villa Urquiza); Las mediciones mensuales comprenden desde junio de 2003 a agosto de 2023.

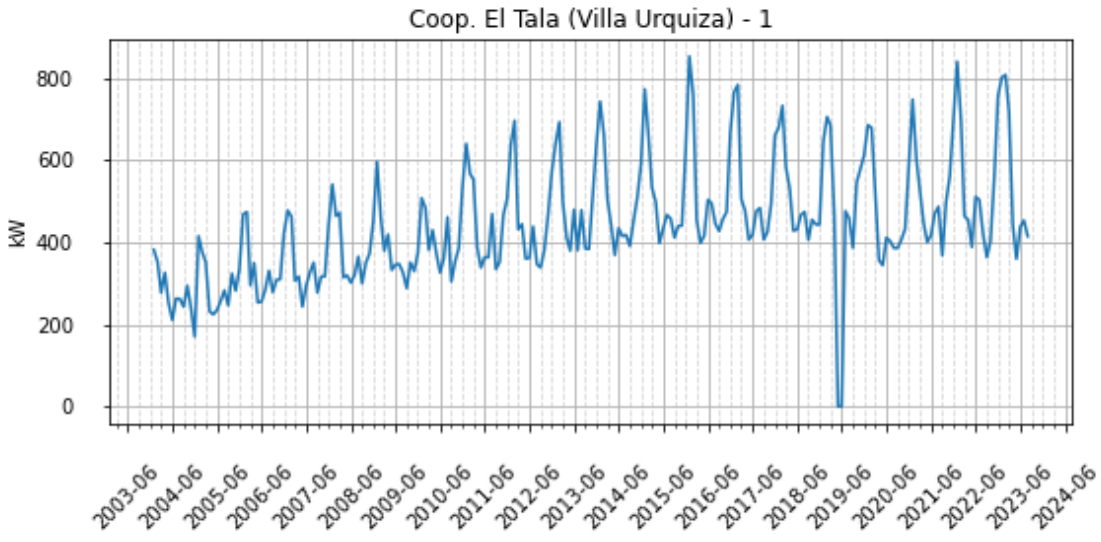

Fig. 20. Demanda de Potencia Activa en [kW] para la localidad de Coop. El Tala (Villa Urquiza).

Ajuste

En la mayoría de los proyectos de aprendizaje profundo, las curvas de entrenamiento y la validación usualmente se visualizan juntas en un gráfico. El propósito de esto es diagnosticar la performance del modelo e identificar qué aspectos necesitan ajuste [6].

La pérdida de entrenamiento (training loss) evalúa el error del modelo en el conjunto de entrenamiento, mientras que la pérdida de validación (validation loss) lo hace en el conjunto de validación, ambos calculando la suma de los errores para cada ejemplo correspondiente. La pérdida de entrenamiento se mide después de cada lote o época y se visualiza mediante una curva, mientras que la pérdida de validación evalúa el rendimiento del modelo en datos no vistos. A continuación, se muestran las curvas correspondientes al ajuste del modelo para esta localidad.

Como se pude observar en la Fig. 21, las curvas de pérdida y validación convergen suavemente. El modelo aprende de los datos, permitiendo que la distancia entre ambas curvas disminuya luego de cada época[7]. Esto representa un buen ajuste, indicando que no hubo sobreajuste (overfitting), es decir, cuando el modelo aprende demasiado bien los datos conocidos pero su performance es mala con respecto a nuevos datos desconocidos; en otras palabras, el modelo no aprende a generalizar.

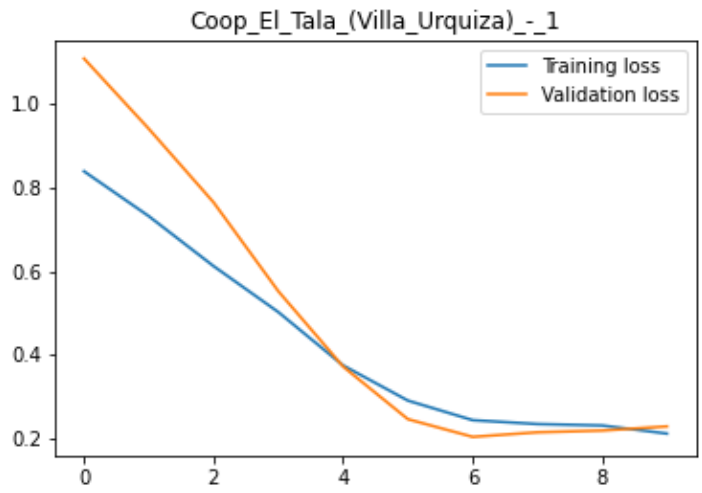

Fig. 21. Curva de ajuste para el modelo entrenado con los datos correspondientes a la localidad de Coop. El Tala (Villa Urquiza). Se visualizan tanto la curva de pérdida en el entrenamiento como en la validación.

Validación

Una vez se tiene el modelo entrenado y se verificó su ajuste, se procede a testearlo. Primero se predice sobre datos conocidos para analizar si la gráfica de la predicción coincide con la de los datos históricos originales. Posteriormente, se le pide al modelo predecir para periodos futuros desconocidos, por fuera de los límites de las mediciones históricas con las que se entrenó el modelo.

En Fig. 22 se puede observar la predicción de la demanda para un periódo de un año, pero comenzando un año antes que el último año de la muestra; es decir, descartando las mediciones del último año como si no existieran. De esta forma, se logra verificar la predicción que genera el modelo con respecto a los datos históricos originales.

De igual manera, en las Fig. 23-26 se visualizan las coincidencias entre la predicción y el histórico para dos, tres, cuatro y diez años.

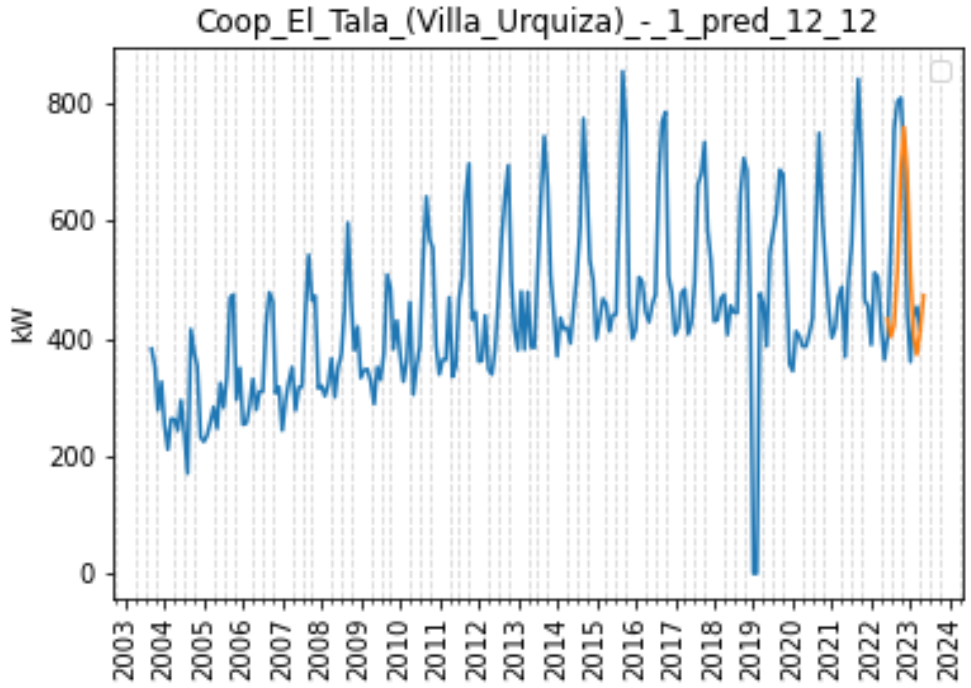

Fig. 22. Predicción a un año desde un año previo a la última muestra para la localidad de Coop. El Tala (Villa Urquiza).

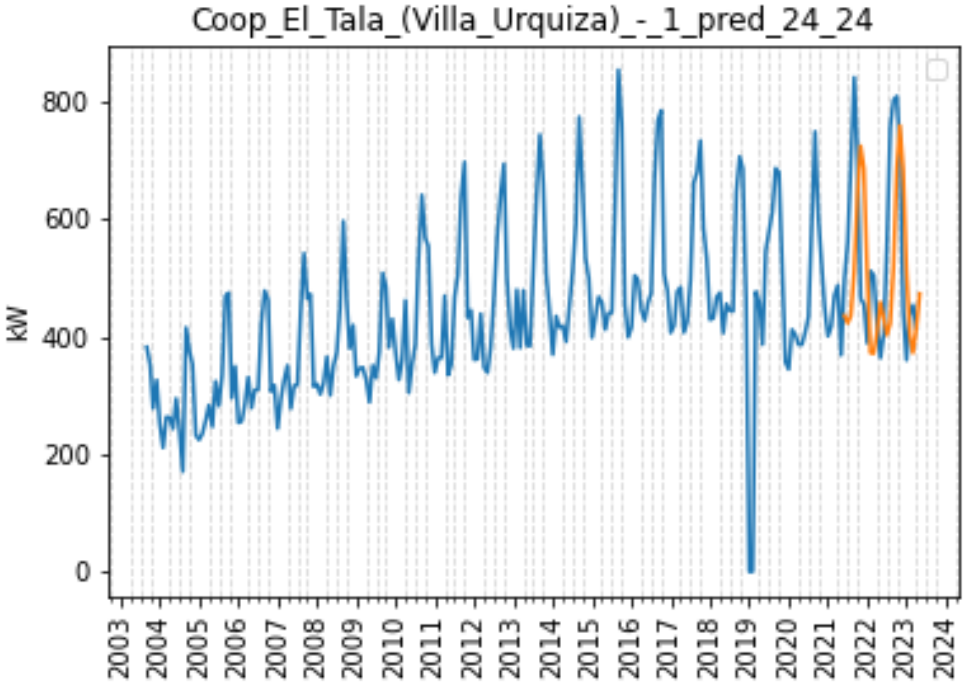

Fig. 23. Predicción a dos años desde dos años previos a la última muestra para la localidad de Coop. El Tala (Villa Urquiza).

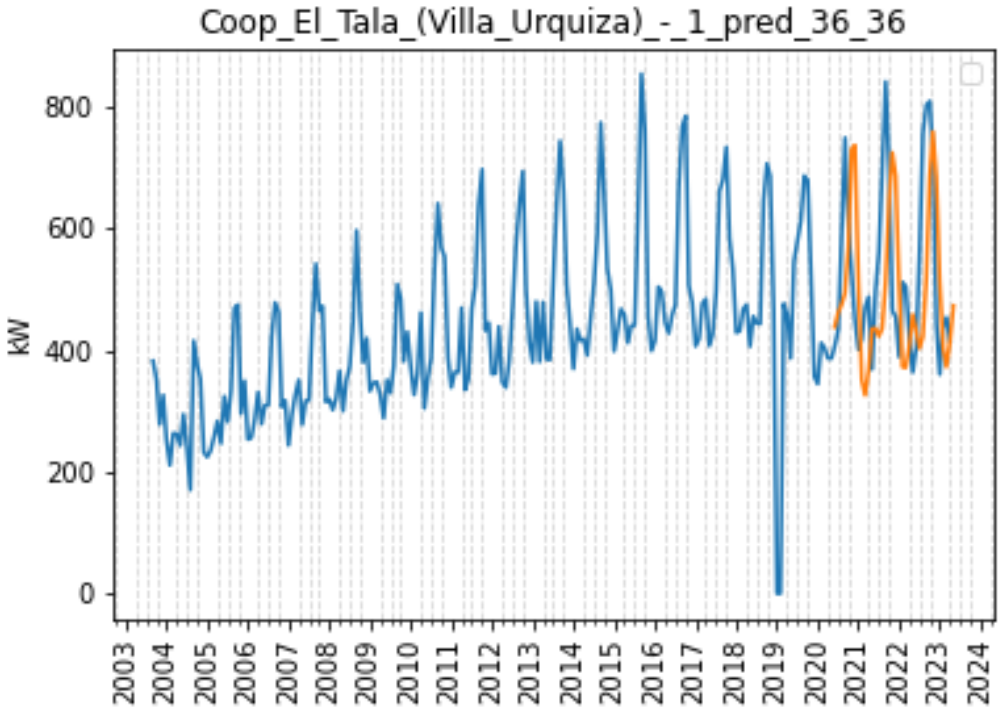

Fig. 24. Predicción a tres años desde tres años previos a la última muestra para la localidad de Coop. El Tala (Villa Urquiza).

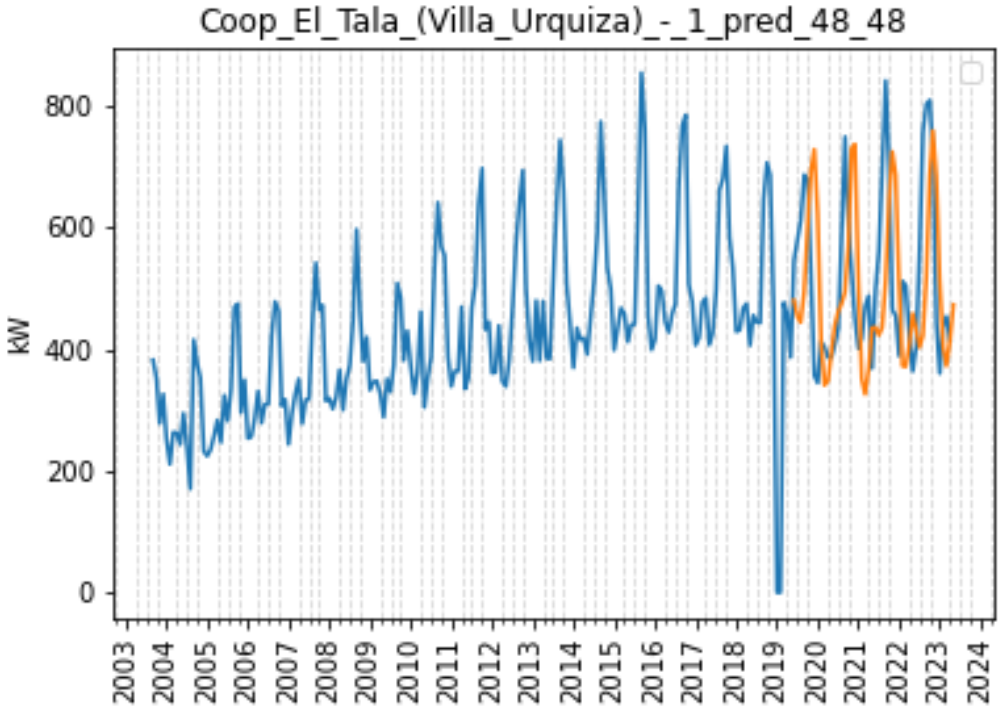

Fig. 25. Predicción a cuatro años desde cuatro años previos a la última muestra para la localidad de Coop. El Tala (Villa Urquiza).

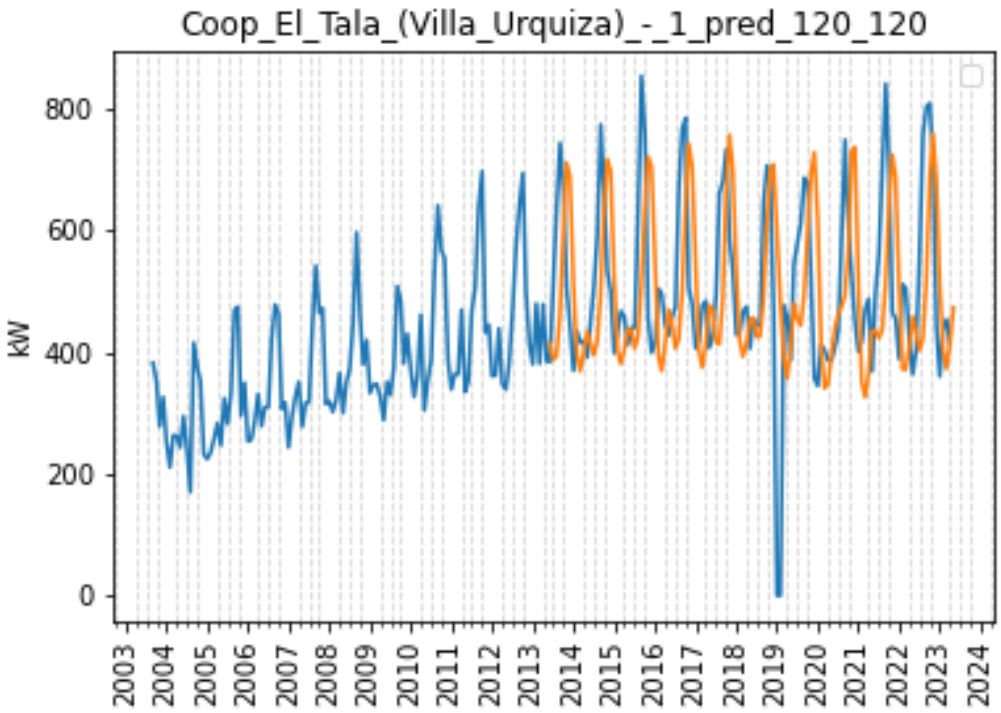

Fig. 26. Predicción a diez años desde diez años previos a la última muestra para la localidad de Coop. El Tala (Villa Urquiza).

Se puede notar que a medida que el modelo se enfrenta a predicciones de mayor alcance, la coherencia de la serie temporal generada tiende a disminuir. Esto es evidente en la Fig. 26, donde, aunque la señal y su tendencia son precisas, estas tienden a simplificarse o suavizarse, y los valores atípicos (outliers) se pierden en el proceso. Es importante destacar el valor extremadamente bajo registrado para el año 2019, lo que resulta en un valle notable en la serie temporal. Este fenómeno puede atribuirse a la ausencia de datos para ciertos períodos o a errores puntuales en la medición. En tal caso, se podría argumentar que el modelo ha predicho correctamente la serie, al haber filtrado este tipo de error al reconocerlo como atípico o anómalo.

Predicciones

En la sección anterior se estudió como el modelo parece haber aprendido a detectar los patrones relevantes y la naturaleza de la serie luego del entrenamiento. Sin embargo, es interesante realizar predicciones fuera de los rangos conocidos. Si bien una predicción no es más que una estimación o inferencia cuyo valor no puede ser comprobado en el presente, se le pide al modelo predecir series temporales a uno, dos, tres y diez años.

Como se visualiza en las Fig. 27-30, los resultados arrojados presentan, al menos, coherencia con la naturaleza de los datos.

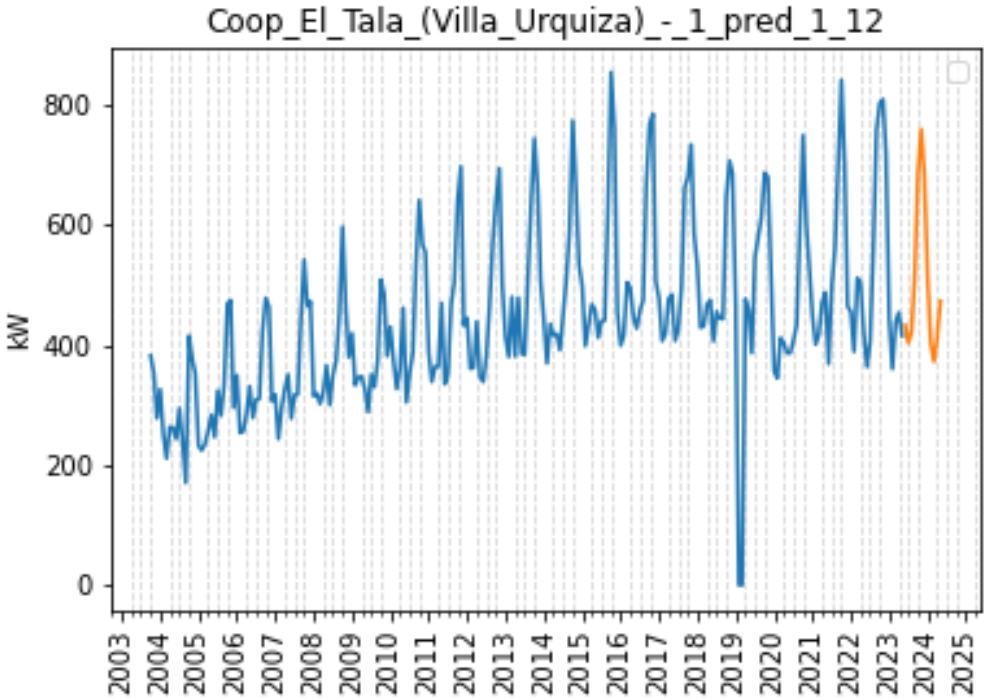

Fig. 27. Predicción a un año posterior a la última muestra para la localidad de Coop. El Tala (Villa Urquiza).

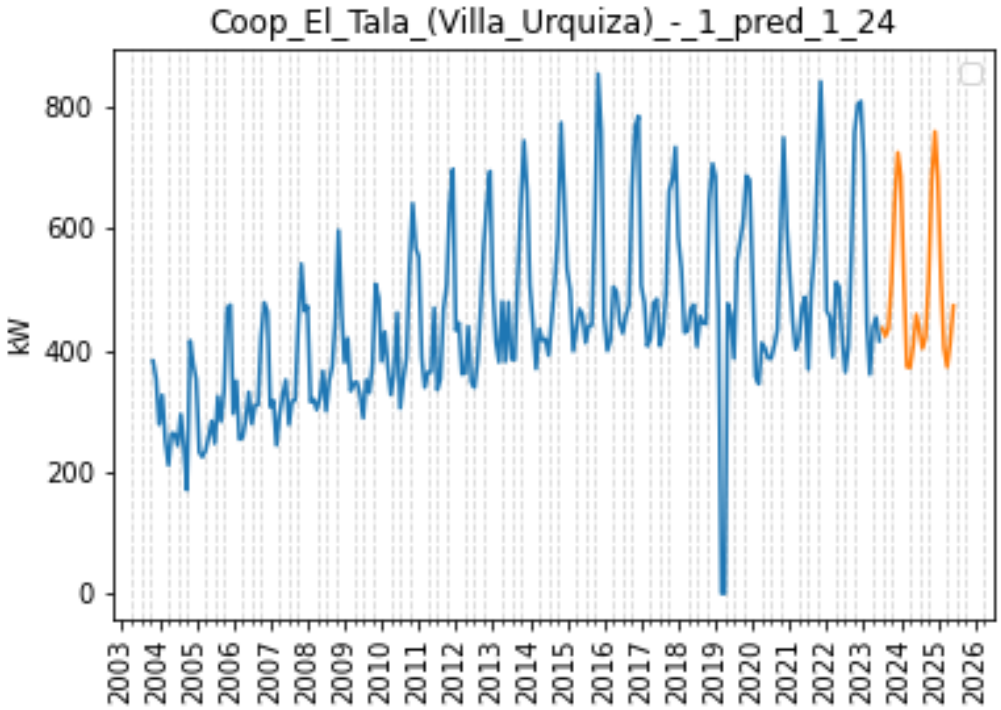

Fig. 28. Predicción a dos años posteriores a la última muestra para la localidad de Coop. El Tala (Villa Urquiza).

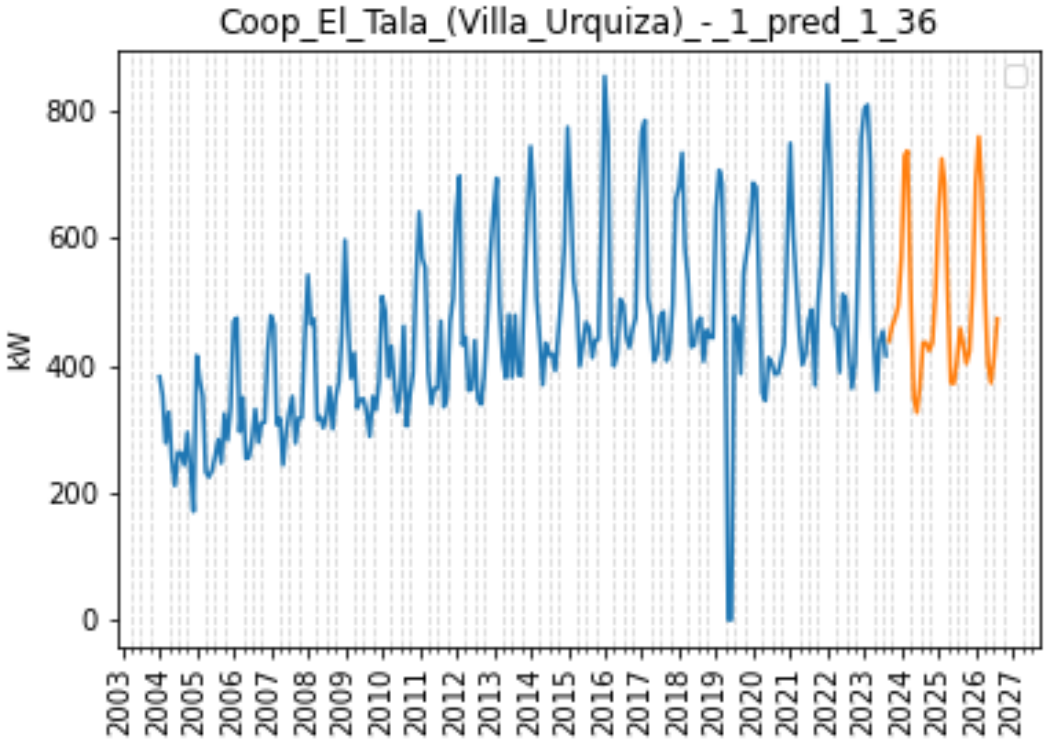

Fig. 29. Predicción a tres años posteriores a la última muestra para la localidad de Coop. El Tala (Villa Urquiza).

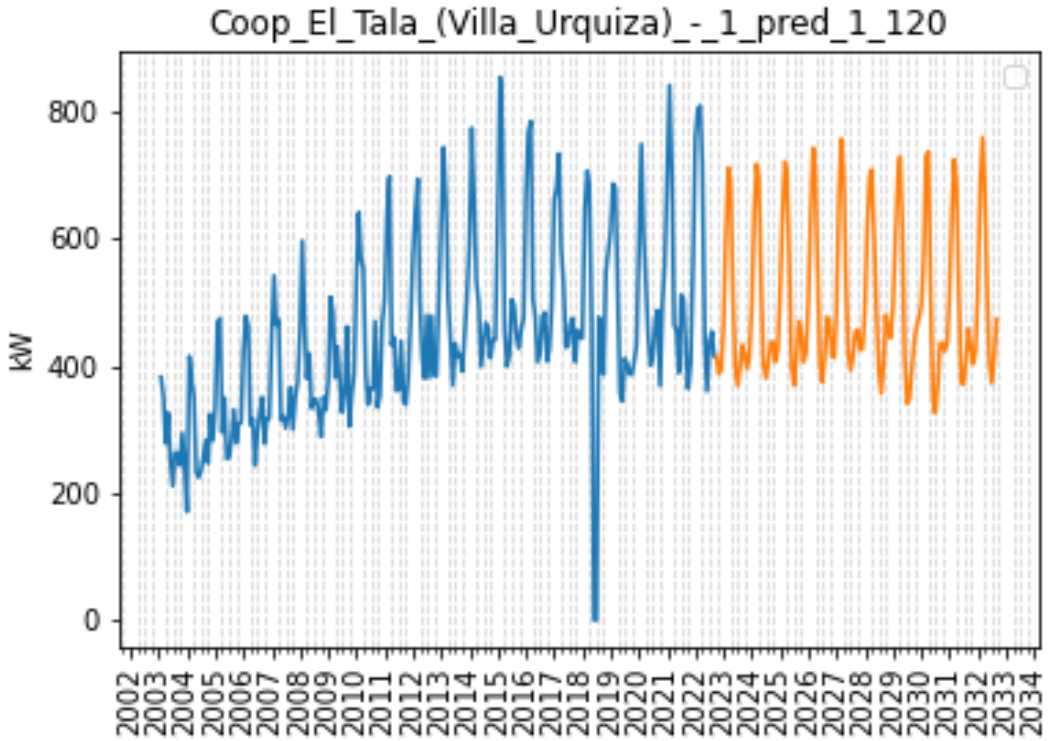

Fig. 30. Predicción a diez años posteriores a la última muestra para la localidad de Coop. El Tala (Villa Urquiza).

4.3.2. Bóvril

A continuación, se replica el análisis realizado en el punto 4.3.1. para la localidad de Bóvril.

En este caso, se observa una serie con mayor irregularidad, caracterizada por múltiples estacionalidades distintas en diferentes períodos. Entre los años 2003 y 2008, seguido por otro ciclo desde 2008 hasta 2011/2012, posteriormente un nuevo ciclo entre 2012 y 2017, y finalmente un descenso que se mantiene dentro de un nuevo rango durante el período 2017-2023. Estos patrones indican una clara variación en la demanda de potencia para dicha localidad.

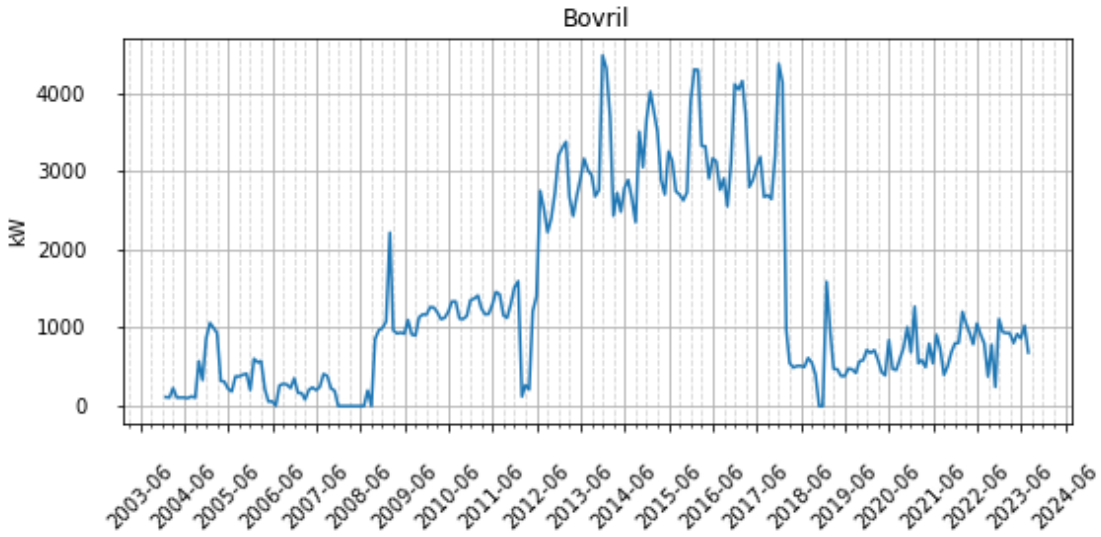

Fig. 31. Demanda de Potencia Activa en [kW] para la localidad de Bóvril.

Ajuste

Al examinar la gráfica de ajuste para esta localidad (Fig. 32), en este caso si bien se observa convergencia, se puede notar que las curvas de pérdida y validación se encuentran más separadas y con un poco más de ruido. Además, la curva de validación tiene a ser plana a lo largo del entrenamiento.

Esto puede indicar la presencia de underfitting (subajuste), lo que significa que el modelo es demasiado simple para capturar la complejidad de los datos subyacentes quizás debido a que el conjunto de datos de entrenamiento no es lo suficientemente representativo o que el modelo no ha sido entrenado adecuadamente. Otra posibilidad es que el conjunto de validación sea mucho más predecible que el conjunto de entrenamiento debido a una posible falta de diversidad en los datos de validación o a una distribución diferente de los datos en comparación con el conjunto de entrenamiento [8]. A modo de comprobación visual, se procede a graficar tanto los resultados de validación como las predicciones para uno, dos, tres y diez años, siguiendo la misma metodología que en el punto anterior.

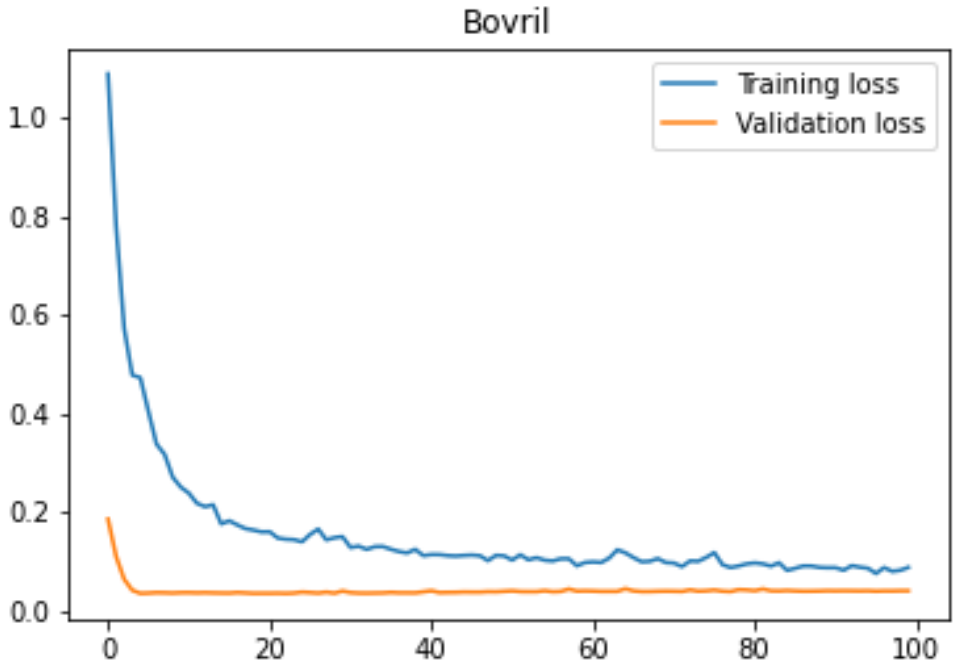

Fig. 32. Curva de ajuste para el modelo entrenado con los datos correspondientes a la localidad de Coop. El Tala (Villa Urquiza). Se visualizan tanto la curva de pérdida en el entrenamiento como en la validación.

Validación

Para la localidad de Bóvril, se interpretan predicciones coherentes, considerando la irregularidad de los datos históricos. Sin embargo, al observar la "copia" de la señal en la Fig. 37 para la predicción a diez años desde diez años previos a la última muestra, se sospecha de overfitting. Esto se debe a que el modelo parece ajustarse demasiado bien a los datos de entrenamiento utilizados para validar, lo cual resulta sospechoso, especialmente dada la naturaleza irregular de las muestras.

Por otro lado, se observan predicciones razonables para validaciones en periodos más cortos."

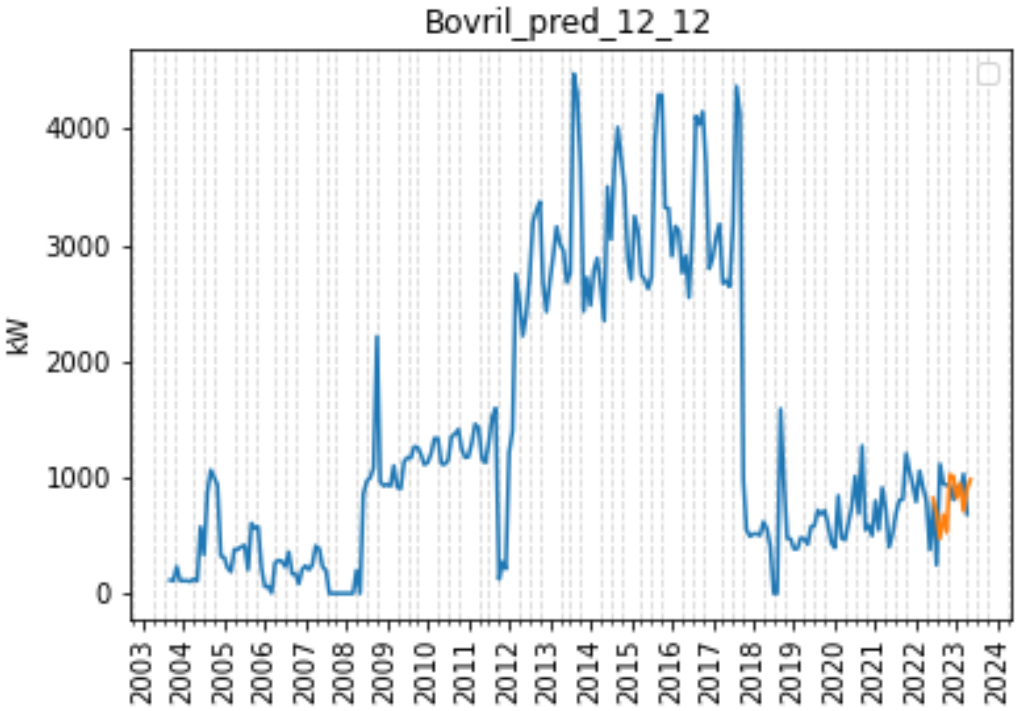

Fig. 33. Predicción a un año desde un año previo a la última muestra para la localidad de Bóvril.

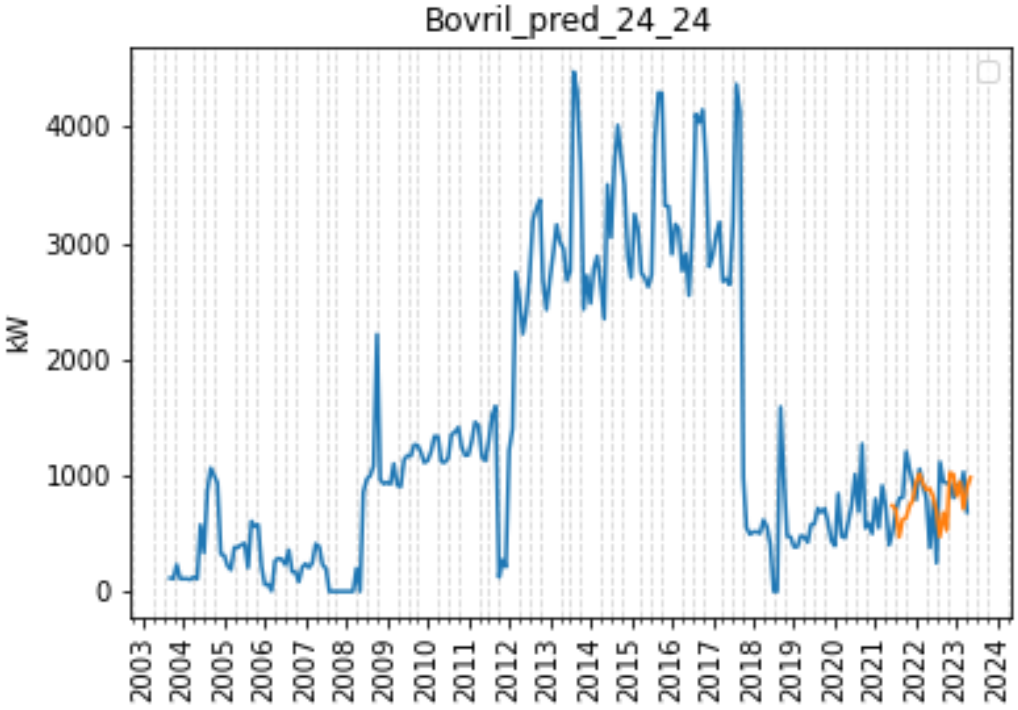

Fig. 34. Predicción a dos años desde dos años previos a la última muestra para la localidad de Bóvril.

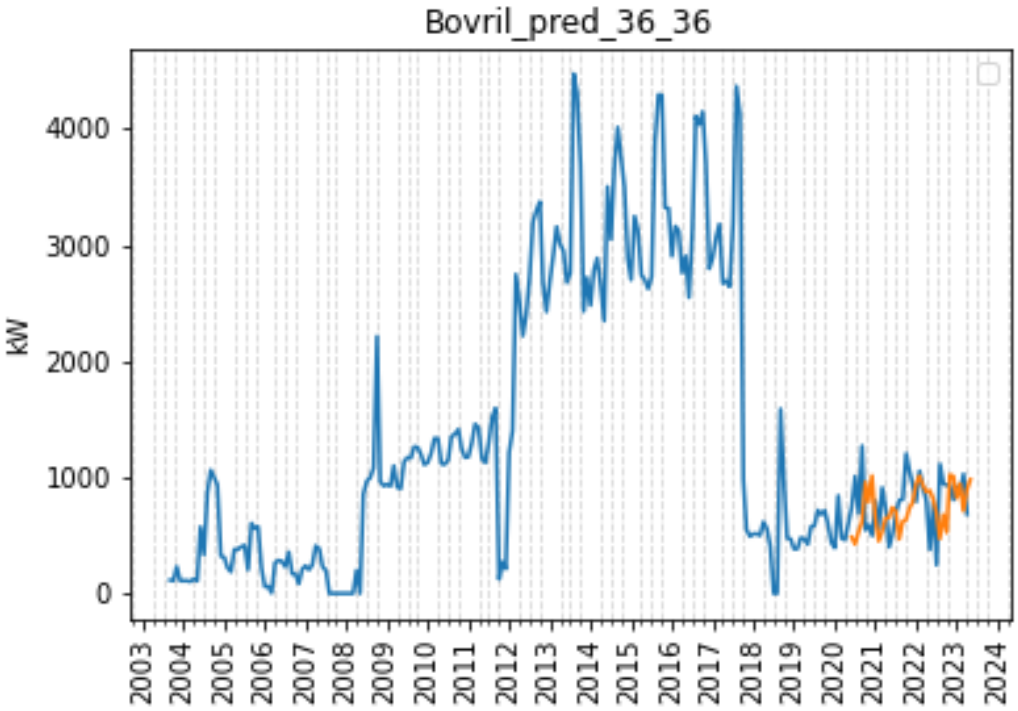

Fig. 35. Predicción a tres años desde tres previos a la última muestra para la localidad de Bóvril.

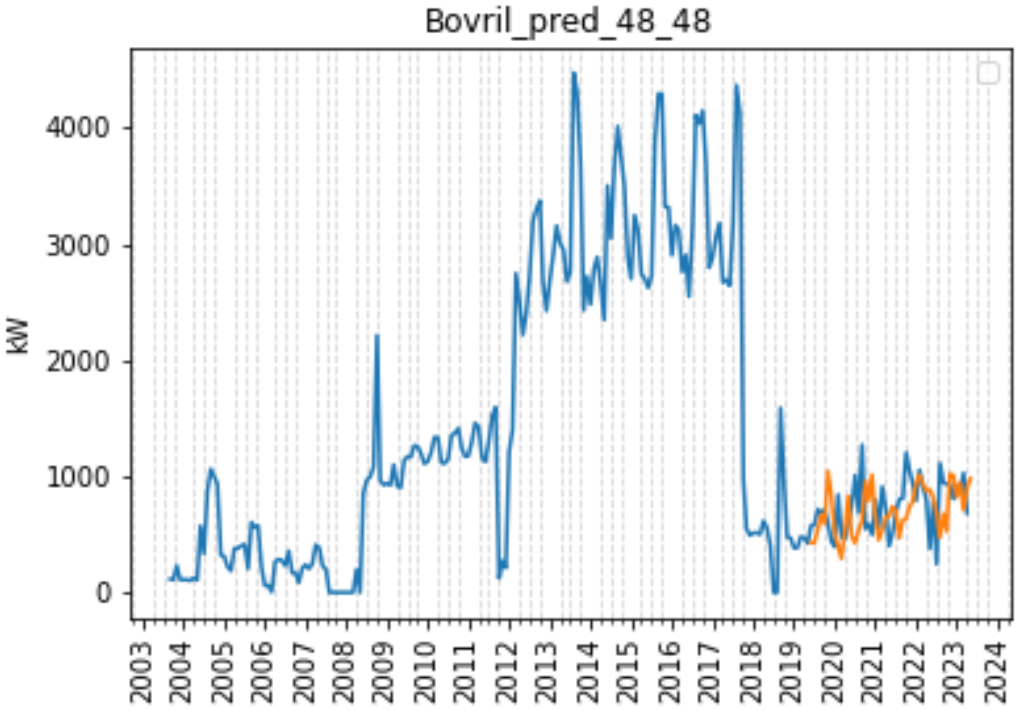

Fig. 36. Predicción a cuatro años desde cuatro años previos a la última muestra para la localidad de Bóvril.

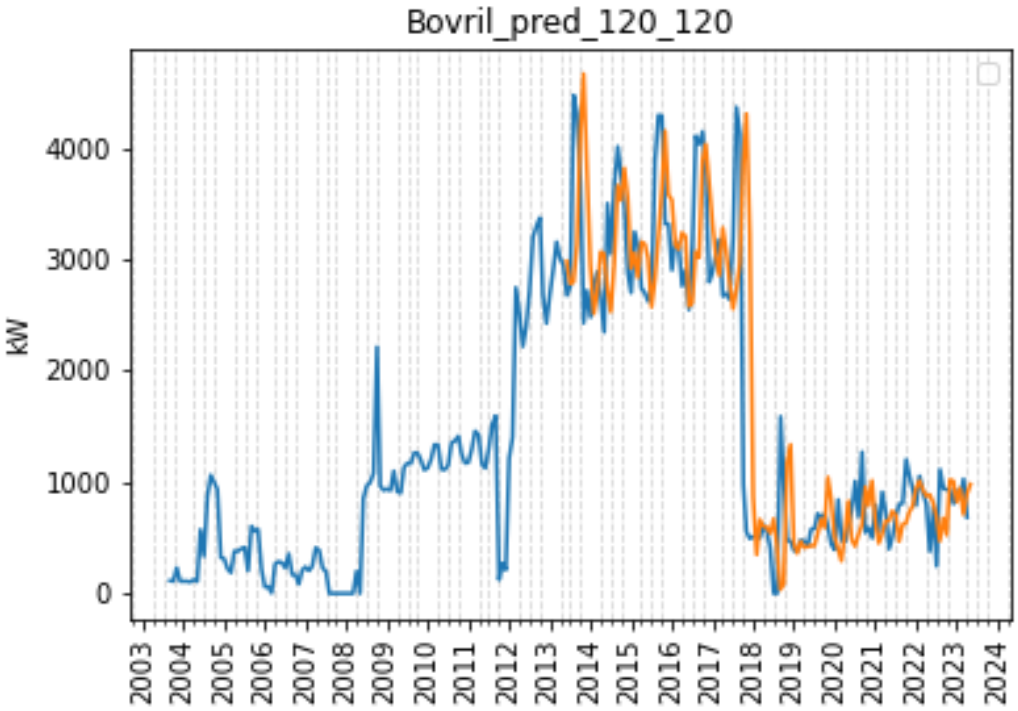

Fig. 37. Predicción a diez años desde diez años previos a la última muestra para la localidad de Bóvril.

Predicciones

Como se evidenció en la validación, para periodos cortos de uno a tres años, la predicción es coherente y razonable (Fig. 38-40). Sin embargo, al realizar una predicción ambiciosa de 10 años (Fig. 41), el resultado es sospechoso, ya que parece replicar el período del 2011 al 2023 como si fuera una nueva onda de iguales características. Esto sugiere que el modelo puede no ser fiable para predicciones tan distantes en el tiempo, y que su entrenamiento pudo haber sido deficiente o que los datos utilizados podrían haber sido poco representativos o escasos.

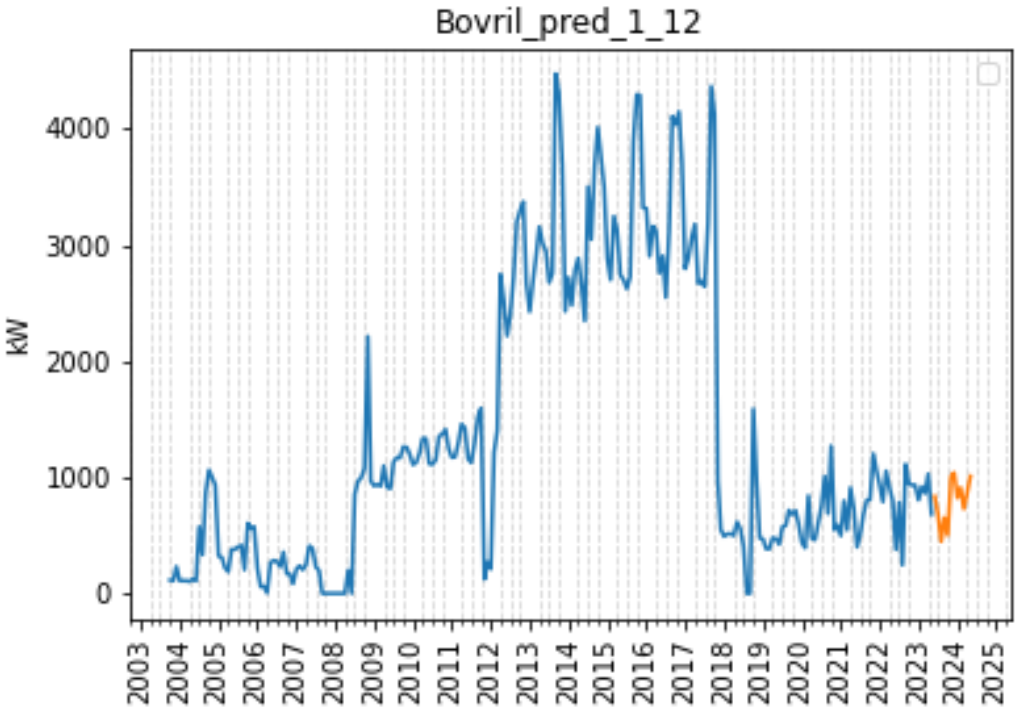

Fig. 38. Predicción a un año posterior a la última muestra para la localidad de Bóvril.

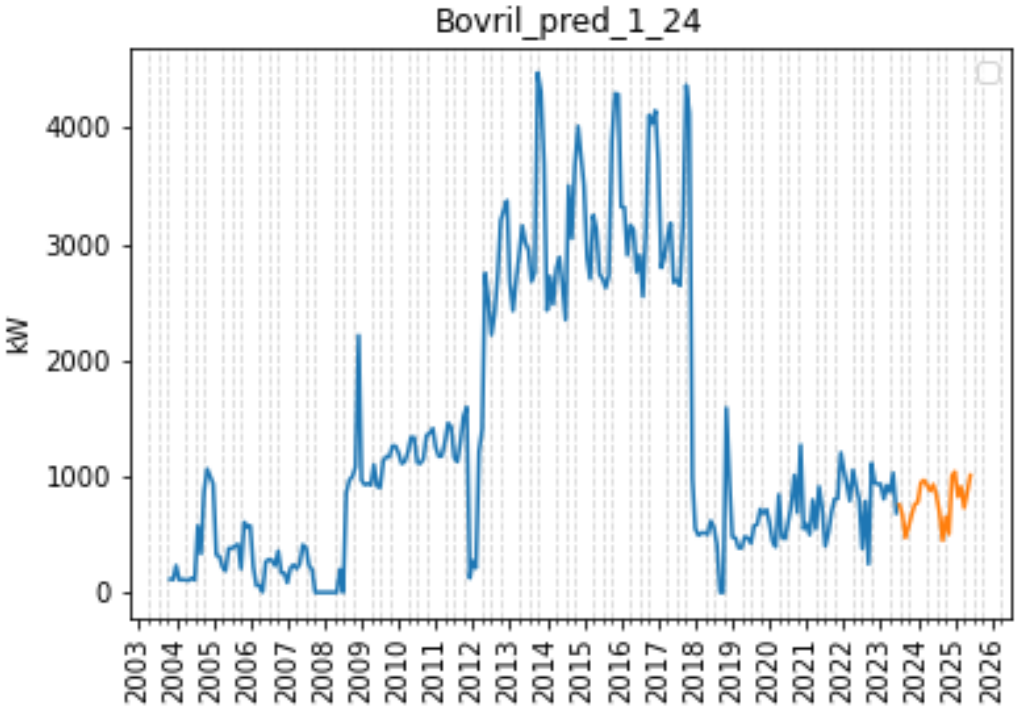

Fig. 39. Predicción a dos años posteriores a la última muestra para la localidad de Bóvril.

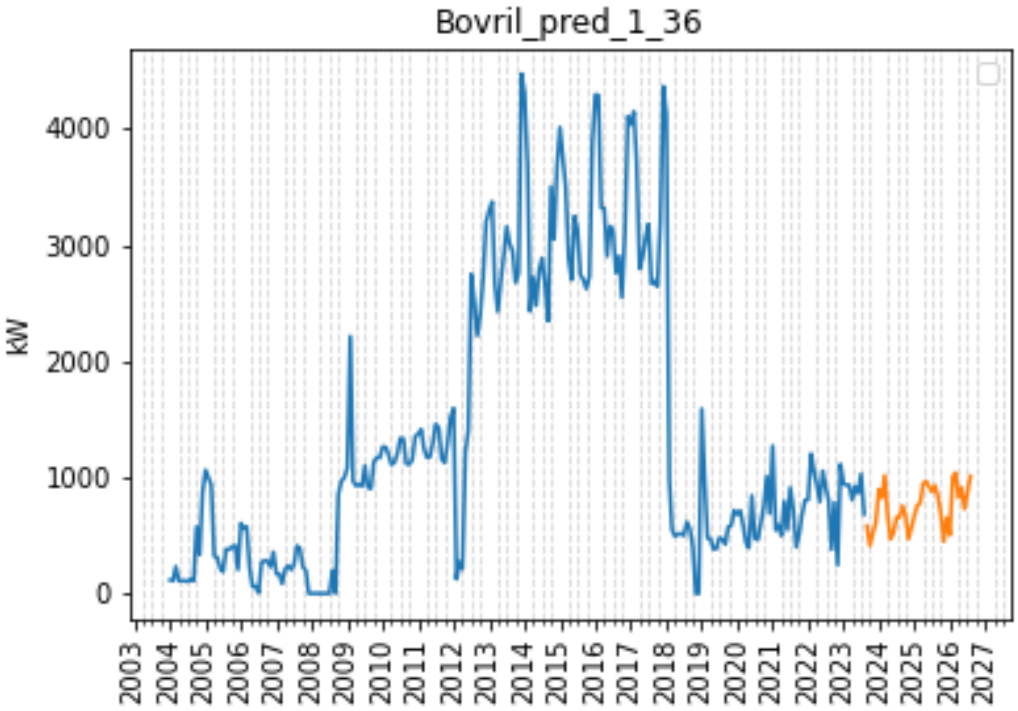

Fig. 40. Predicción a tres años posteriores a la última muestra para la localidad de Bóvril.

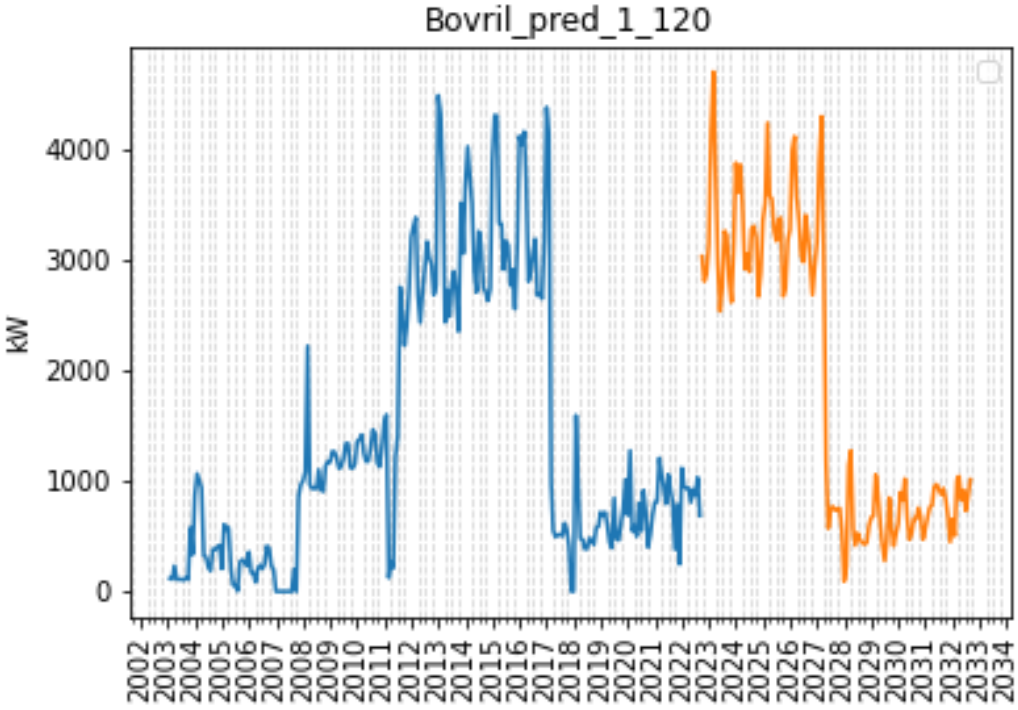

Fig. 41. Predicción a diez años posteriores a la última muestra para la localidad de Bóvril.

4.4. Conclusión

Al analizar los resultados obtenidos tanto en la validación frente a datos conocidos como en las predicciones proyectadas en el tiempo, se observa que los modelos entrenados logran capturar cierta estacionalidad, es decir, detectan patrones o ciclos recurrentes que se presentan regularmente en el conjunto de datos a lo largo del tiempo. Sin embargo, se evidencia una disminución en la precisión a medida que la predicción se extiende en el tiempo. Esta reducción en la precisión puede atribuirse a la complejidad de anticipar eventos a largo plazo o a un entrenamiento insuficiente, así como a la falta de representatividad en el conjunto de datos utilizado.

Por otra parte, al comparar las predicciones con los datos históricos conocidos, se observa un ajuste preciso en las gráficas, casi replicando los valores originales. Este ajuste sugiere la posibilidad de overfitting o sobreajuste, indicando que el modelo podría haber aprendido de manera demasiado específica los datos de entrenamiento. Aunque las curvas de ajuste convergen, esta convergencia podría estar influenciada por la limitada disponibilidad del conjunto de datos original.

A pesar de estos desafíos, los resultados para predicciones a corto plazo son coherentes y razonables. En general, ambos casos presentan desafíos y hallazgos positivos, concluyendo el estudio de manera satisfactoria y destacando perspectivas alentadoras para la aplicación de estos modelos en la predicción de la demanda.

Para futuras investigaciones, se recomienda realizar un análisis detallado de diversas arquitecturas de capas y explorar variaciones en los hiperparámetros, como la cantidad de neuronas, funciones de activación, optimización y pérdida. Además, se resalta la importancia de contar con conjuntos de datos más extensos y con una mayor frecuencia de muestreo, como la recolección de datos por hora o minuto en lugar de mensualmente. Se sugiere explorar otros enfoques de entrenamiento, como el "transfer learning", donde el modelo se reentrena con conjuntos de datos diferentes, y considerar modelos multivariados que integren series temporales de otras variables además de la potencia.

5. Bibliografía

1. Nielsen, M. (2015). Neural Networks and Deep Learning. Determination Press.
2. Olah, C. (2015). Understanding LSTMs Networks. Recuperado de https://colah.github.io/posts/2015-08-Understanding-LSTMs.
3. Shalev-Shwartz, S., & Ben-David, S. (2014). Understanding Machine Learning: From Theory to Algorithms. Nueva York, Estados Unidos: Cambridge University Press.
4. IBM. ¿Qué son las redes neuronales recurrentes?. Recuperado de https://www.ibm.com/mx-es/topics/recurrent-neural-networks#:~:text=Una%20red%20neuronal%20recurrente%20(RNN,voz%20y%20subt%C3%ADtulos%20de%20im%C3%A1genes.
5. Gomez, A. (2016). Backpropagating an LSTM: A Numerical Example. Recuperado de https://medium.com/@aidangomez/let-s-do-this-f9b699de31d9.
6. Baeldung. (2023). Training and Validation Loss in Deep Learning. Recuperado de https://www.baeldung.com/cs/training-validation-loss-deep-learning.
7. Soleymani, A. (2022). Your validation loss is lower than your training loss? This is why! Towards Data Science. Recuperado de https://towardsdatascience.com/what-your-validation-loss-is-lower-than-your-training-loss-this-is-why-5e92e0b1747e.
8. Brownlee, J. (2019). Machine Learning Mastery. How to use Learning Curves to Diagnose Machine Learning Model Performance. Recuperado de https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/.
9. Weber, J. (2023). Copy Programming. Why the 6 in relu6? Recuperado de https://copyprogramming.com/howto/why-the-6-in-relu6.
10. Brownlee, J. (2017). Machine Learning Mastery. Long Short-Term Memory Networks WITH PYTHON.
11. Box, G. E. P., Jenkins, G. M., et al. (2015). Time Series Analysis: Forecasting and Control. Wiley.