Universidad Politécnica
de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos

Grado en Ingeniería Informática

Trabajo Fin de Grado

Memoria de Seguimiento

Aplicación de Redes Neuronales en la Predicción de Series Temporales de Acciones Bursátiles

Autor: Marc Rubí Reus

Tutor: Ángel Mario García Pedrero

Madrid, junio 2025

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado
Grado en Ingeniería Informática
Título: Aplicación de Redes Neuronales en la Predicción de Series Temporales de Acciones Bursátiles
Junio 2025

Autor: Marc Rubí Reus
Tutor:
Ángel Mario García Pedrero
Departamento de Arquitectura y Tecnología de Sistemas Informáticos
ETSI Informáticos
Universidad Politécnica de Madrid

Resumen

Este trabajo de fin de grado aborda la implementación y comparación de cuatro arquitecturas de redes neuronales aplicadas a la predicción de series temporales bursátiles: Echo State Network (ESN), Kolmogorov-Arnold Network (KAN), Long Short-Term Memory (LSTM) y Liquid Neural Network (LNN). Se emplean precios diarios ajustados de las diez empresas con mayor relevancia en el índice S&P 500, utilizando tres ventanas históricas diferentes (aproximadamente 5, 10 y 13 años).

Cada modelo se entrena bajo las mismas condiciones metodológicas, incluyendo la normalización Min-Max, creación de secuencias mediante una ventana deslizante, optimizador Adam, función de coste MSE, parada temprana y ajustes dinámicos en la tasa de aprendizaje. Posteriormente, se evalúa el rendimiento predictivo mediante seis métricas: MSE, RMSE, MAE, MAPE, SMAPE y R^2, complementando el análisis con gráficas que muestran el ajuste del modelo frente a los datos reales, así como la evolución del error durante el entrenamiento.

Finalmente, se realiza una comparación general y detallada entre los modelos, examinando cómo influyen en los resultados factores como la volatilidad específica de cada empresa y la longitud de la ventana histórica. Adicionalmente, se plantean posibles líneas futuras para continuar con esta investigación.

Abstract

This final degree project focuses on the implementation and comparative evaluation of four neural network architectures for stock-price time series forecasting: Echo State Network (ESN), Kolmogorov-Arnold Network (KAN), Long Short-Term Memory (LSTM), and Liquid Neural Network (LNN). The models are trained using adjusted daily prices from the top ten companies in the S&P 500, considering three different historical windows (approximately 5, 10, and 13 years).

All models follow an identical methodological setup, including Min-Max normalization, sequence generation through a sliding window, the Adam optimizer, Mean Squared Error loss function, early stopping criteria, and adaptive learning rate adjustments. Predictive performance is assessed with six metrics: MSE, RMSE, MAE, MAPE, SMAPE, and R^2. The analysis is complemented with visual representations showing model predictions compared to actual data, as well as training and validation error curves.

Lastly, a detailed comparison among the four architectures is presented, discussing how factors such as individual stock volatility and historical window length affect the prediction quality. Potential future research directions are also proposed.

Tabla de contenidos

1 Introducción

1.1 Contexto y relevancia de las series temporales

Dentro del mundo de las finanzas, anticiparse a lo que puede pasar en el mercado es fundamental. Para ello, se suele recurrir a las series temporales. Estos son datos ordenados en el tiempo que sirven para analizar cómo evolucionan cosas como el precio de una acción o el volumen de transacciones a lo largo de los días, meses o años [1].

Este tipo de análisis no solo ayuda a entender lo que ya ha pasado, sino que permite hacer predicciones sobre lo que puede venir, algo clave en un entorno tan cambiante como el financiero [1]. Si se detectan bien ciertos patrones, estacionalidades o cambios bruscos, es posible adelantarse a ciertos cambios antes de que el resto de inversores los detecten. Por eso, las series temporales están detrás de muchos modelos de predicción bursátil, alertas automáticas o incluso algoritmos de inversión [1].

Sin embargo, predecir con series temporales financieras no es fácil. Estos datos suelen tener mucho ruido, no son lineales y su comportamiento puede cambiar de un momento a otro. Eso hace que los modelos clásicos como *ARIMA* se queden cortos, ya que asumen que los datos son estables y siguen relaciones simples [6].

En los últimos años, han ganado terreno los modelos basados en redes neuronales, sobre todo los más modernos. Esto ha sido debido a que son capaces de captar mejor la complejidad de los datos financieros, adaptarse a su variabilidad y encontrar patrones que no son tan evidentes a simple vista [2][4].

1.2 Aplicaciones de las series temporales

Aunque el uso de series temporales se da en muchos sectores, es en el ámbito financiero donde más protagonismo han tenido. En esta área, se usan para intentar predecir precios futuros, detectar riesgos o afinar estrategias de inversión [1][3].

Gracias a su capacidad para reflejar el comportamiento de activos, índices o variables económicas, han motivado el desarrollo de técnicas más potentes que las estadísticas tradicionales. Dentro de estas nuevas técnicas, las redes neuronales no convencionales están cobrando protagonismo, especialmente cuando se buscan modelos más flexibles, precisos y adaptables [2].

1.3 Objetivos del proyecto

Predecir precios bursátiles es uno de los mayores retos que hay dentro del análisis de datos. Esto ocurre no solo por lo inestable que es el mercado, sino también por la gran cantidad de factores externos que pueden influir. En este contexto, los métodos clásicos como *ARIMA (AutoRegressive Integrated Moving Average)* o *GARCH* (*Generalized Autoregressive Conditional Heteroskedasticity*) tienen limitaciones claras. Las redes neuronales, por el contrario, están demostrando ser una alternativa más potente, sobre todo por su capacidad para aprender relaciones complejas y adaptarse al comportamiento cambiante del mercado [2][4][5].

El objetivo de este Trabajo de Fin de Grado es evaluar cómo funcionan cuatro arquitecturas de redes neuronales no convencionales cuando se aplican a series temporales financieras. Para ello, se compararán sus resultados usando datos reales y se analizarán métricas cuantitativas de error (MSE, RMSE, MAE, R^2, MAPE/SMAPE), coste computacional (tiempo de entrenamiento y número de parámetros), estabilidad en la convergencia (oscilaciones en la pérdida de validación) y capacidad de generalización en datos no vistos.

1.3.1 Objetivo general

Analizar y comparar cuatro arquitecturas de redes neuronales no convencionales (Echo State Networks ESN, Kolmogorov-Arnold Networks KAN, Long Short-Term Memory Networks LSTM y Liquid Neural Networks LNN) en su capacidad para predecir precios de acciones bursátiles a corto y medio plazo.

1.3.2 Objetivo específico

Los objetivos específicos de este trabajo incluyen estudiar en profundidad los fundamentos teóricos de las redes neuronales ESN, KAN, LSTM y LNN. Asimismo, se busca implementar estas arquitecturas para predecir series temporales financieras a partir de datos históricos de precios bursátiles, llevando a cabo un experimento comparativo que permita evaluar su rendimiento bajo condiciones homogéneas. Se pretende entrenar cada uno de los modelos y ajustar adecuadamente sus hiperparámetros con el fin de maximizar su precisión y capacidad predictiva. Finalmente, se realizará una evaluación completa mediante métricas habituales en el ámbito financiero, como precisión, eficiencia computacional y generalización, concluyendo con un análisis crítico de los resultados y planteando posibles líneas de mejora o recomendaciones para futuros estudios.

2 Marco teórico

2.1 Fundamentos de series temporales en finanzas

2.1.1 Características y comportamiento de las series financieras

Las series temporales financieras, como las que representan precios de acciones, tipos de interés o volúmenes de negociación, se caracterizan por una serie de propiedades estadísticas y estructurales que las distinguen de otros tipos de series. Estas características tienen un impacto directo en su modelado y predicción, por lo que su análisis resulta esencial como paso previo al diseño de soluciones basadas en redes neuronales.

No estacionariedad

Una de las principales propiedades de las series temporales financieras es su no estacionariedad, lo que significa que sus propiedades estadísticas, como la media o la varianza, cambian a lo largo del tiempo [6]. Esta característica implica que las técnicas que asumen una distribución estable de los datos no son adecuadas sin una transformación previa, como la diferenciación temporal o la transformación logarítmica. La no estacionariedad está asociada a cambios estructurales en el mercado, noticias inesperadas, ciclos económicos, entre otros factores.

En la Figura 1 se muestra cómo el uso de una media móvil (promedio de los precios de los últimos N días para suavizar la serie) puede ayudar a identificar la dirección de la tendencia en los precios de una acción, incluso cuando hay fluctuaciones a corto plazo.

Figura 1. Media móvil aplicada a precios bursátiles [7].

Volatilidad cambiante (heterocedasticidad)

Las series de precios bursátiles presentan episodios de alta y baja volatilidad, que no se distribuyen de forma constante. Este fenómeno, conocido como heterocedasticidad, ha sido ampliamente estudiado y modelado a través de modelos como *Autoregressive Conditional Heteroskedasticity (ARCH)* o *General Autoregressive Conditional (GARCH)* [8]. En la práctica, esto significa que los errores de predicción también tienden a agruparse en periodos de alta incertidumbre, dificultando la estimación de intervalos de confianza estables.

Tendencias y ciclos

Pese a su aparente aleatoriedad, muchas series bursátiles muestran tendencias a largo plazo (por ejemplo, crecimiento sostenido de un índice) o ciclos (repetición de comportamientos a distintos intervalos), que pueden estar relacionados con la estacionalidad del mercado, políticas económicas o efectos psicológicos del comportamiento inversor [9].

En la Figura 2 se ilustra una descomposición *Seasonal-Trend Losses* (*STL*) un método que emplea regresión local para separar los componentes principales de una serie temporal: tendencia, estacionalidad y ruido.

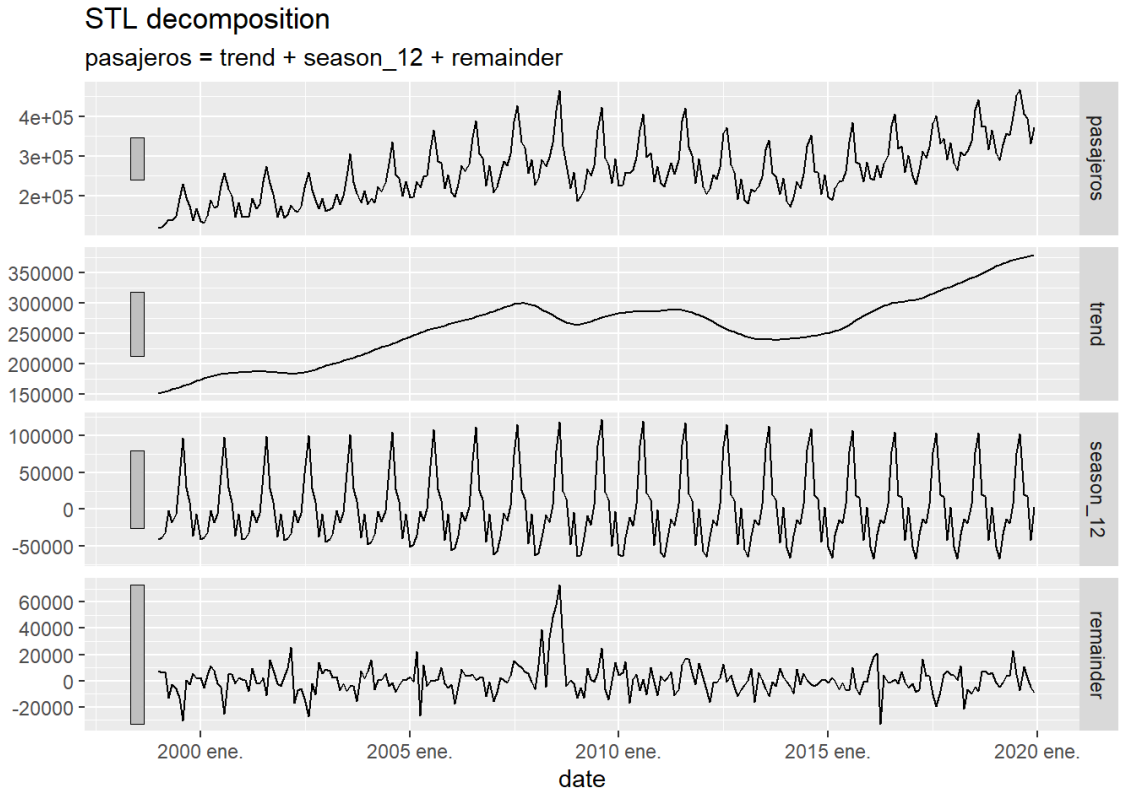

Figura 2. Descomposición STL de una serie temporal [10].

Autocorrelación y memoria temporal

En las series bursátiles es común que los valores pasados influyan sobre los futuros, especialmente en horizontes de corto plazo. Este fenómeno se conoce como autocorrelación, y su presencia puede aprovecharse mediante modelos que capturan relaciones temporales entre observaciones, como los modelos AR (Autorregresivo) o las redes recurrentes [11].

Influencia de factores exógenos

Las series financieras están fuertemente condicionadas por factores externos: anuncios macroeconómicos, resultados empresariales, decisiones políticas, entre otros. Estos eventos pueden provocar variaciones abruptas e impredecibles en la serie, añadiendo una capa extra de complejidad a su modelado [12].

2.1.2 Métodos tradicionales de análisis y predicción

Durante muchos años el análisis y la predicción de series financieras en el mercado bursátil se han abordado mediante métodos estadísticos [4]. Estos modelos matemáticos se han utilizado tanto en la predicción de series bursátiles como en otros ámbitos. Entre ellos se encuentran modelos como *AR, MA, ARMA ARIMA* [4], así como también *GARCH* [8]. Estos se basan en relaciones matemáticas simples entre observaciones pasadas y actuales. Han demostrado

que funcionan especialmente si los datos son estacionarios o tienen estructuras lineales [4].

En el ámbito financiero se han utilizado sobre todo para la predicción de precios, la estimación de la volatilidad o la detección de ciclos económicos [4]. El problema que se ha encontrado con estos modelos ha sido la volatilidad de las series bursátiles, lo que ha hecho que se busquen modelos más robustos [5].

A continuación, se presentan los principales modelos utilizados en la predicción de series temporales, junto con una breve descripción de su funcionamiento y limitaciones.

AR, MA, ARMA y *ARIMA*

Estos modelos comparten la característica de que se basan en relaciones lineales entre los datos pasados y los valores actuales de la serie.

Estos modelos comparten como característica principal el basarse en relaciones lineales entre los valores anteriores y el actual de la serie. El modelo Autorregresivo (AR) utiliza valores pasados de la serie para predecir el valor actual, siendo apropiado cuando existe una autocorrelación clara en los datos. El modelo de Media Móvil (MA), por su parte, predice el valor actual a partir de los errores aleatorios generados en observaciones anteriores. La combinación de ambos modelos se materializa en el ARMA (AutoRegressive Moving Average), especialmente indicado para series estacionarias. Finalmente, el modelo ARIMA (AutoRegressive Integrated Moving Average) amplía el modelo ARMA con un proceso de diferenciación adicional, lo cual permite abordar series no estacionarias, aunque presenta limitaciones al captar relaciones lineales más complejas [4].

SARIMA

El modelo *SARIMA (Seasonal AutoRegressive Integrated Moving Average*) es una extensión del modelo ARIMA ya mencionado anteriormente. La diferencia que tiene con ese modelo es que su uso es únicamente para series con estacionalidad ya que usa componentes estacionales: diferencia estacional y parte autorregresiva estacional. No es el modelo más común ya que en las series bursátiles no siempre hay una estacionalidad clara [4].

ARCH y GARCH

Los modelos *ARCH* (*Autoregressive Conditional Heteroskedasticity*) y *GARCH* (*Generalized Autoregressive Conditional Heteroskedasticity*) son modelos que se utilizan específicamente para series donde la varianza cambia con el tiempo, como en el caso de las acciones bursátiles. La principal diferencia que presentan con el modelo ARIMA es que estos modelos no predicen el precio de la acción, sino su dispersión. Esto hace que sean ampliamente utilizados para calcular medidas como Value At Risk. Sin embargo, su enfoque lineal supone una limitación estructural importante, ya que les impide modelar de forma adecuada las dinámicas no lineales y las interacciones complejas entre múltiples factores que caracterizan a las series temporales en entornos financieros.

A pesar de que estos modelos se han utilizado durante años para la predicción bursátil, presentan limitaciones para captar el comportamiento real de las series financieras. Esto se debe a que asumen relaciones lineales y no se adaptan bien a cambios estructurales ni a interacciones complejas no evidentes. Estas carencias justifican la exploración de enfoques más flexibles, como las redes neuronales, que se tratarán en el siguiente apartado [8].

2.2 Redes neuronales en la predicción de series temporales

Tras evidenciar las limitaciones que presentan los modelos estadísticos clásicos, surge la necesidad de desarrollar modelos más flexibles y capaces de capturar relaciones no lineales como las redes neuronales [8]. A diferencia de modelos como ARIMA o GARCH, las redes neuronales no necesitan asumir que los datos siguen patrones lineales o estacionales.

Las redes neuronales son altamente eficaces en la predicción de series bursátiles, ya que pueden detectar patrones ocultos y adaptarse a dinámicas con tendencias variables [2]. También están diseñadas para tratar con secuencias (como LSTM, ESN, etc.) y se utilizan cada vez más en la predicción bursátil, ya que son capaces de manejar la complejidad y volatilidad del mercado. Sin embargo, presentan ciertas limitaciones como el alto coste computacional, el riesgo de *overfitting* y la necesidad de ajustar los hiperparámetros para obtener buenos resultados [4].

En este trabajo se analizarán 4 arquitecturas concretas para la predicción de series temporales bursátiles: ESN, KAN, LSTM, LNN.

2.2.1 *Echo State Network* (ESN)

2.2.1.1 Arquitectura y funcionamiento

Las Redes Neuronales Recurrentes (RNN) son un tipo de arquitectura diseñada para trabajar con secuencias de datos, ya que permiten que la salida actual dependa tanto de la entrada como del estado anterior. Esto les proporciona una memoria temporal, lo que las hace especialmente útiles en tareas como la predicción de series temporales. Sin embargo, al entrenarse con retropropagación a través del tiempo [13], presentan problemas como el desvanecimiento o la explosión del gradiente cuando las secuencias son largas.

ESN es un tipo de Red Neuronal Recurrente (RNN), y pertenece a una familia de modelos llamados *Reservoir Computing.* Esta red fue propuesta para solucionar problemas de estabilidad que tenían las RNN tradicionales [14].

La arquitectura ESN consta de tres partes principales, la entrada (*input layer*), el reservorio dinámico o capa oculta y la capa de salida (*output layer*). Esta estructura puede verse en la Figura 3, donde se observa cómo las conexiones internas del reservorio permanecen fijas y solo se entrena la capa de salida. Las conexiones retroalimentadas permiten incorporar la influencia de salidas pasadas [14].

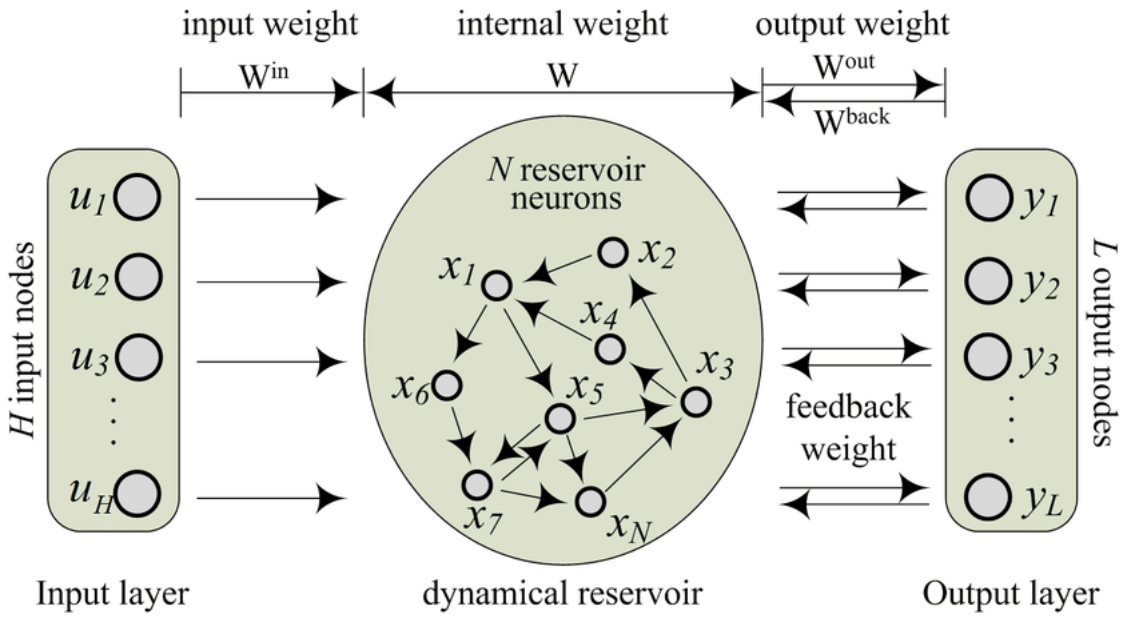

Figura 3. Arquitectura de una Echo State Network (ESN) [15].

De estas tres partes, la única que se entrena es la capa de salida. Las conexiones del reservorio en vez de entrenarse se inician aleatoriamente y se mantienen fijas. Las ventajas que ofrece esta red son múltiples: es mucho más eficiente que entrenar una RNN completa, tiene buena capacidad con dependencias temporales y secuencias largas y evita problemas que son típicos en el entrenamiento de las RNN como el desvanecimiento del gradiente.

Una característica fundamental de las *Echo State Networks* es la denominada *Echo State Property* (ESP), la cual garantiza que el efecto de las entradas pasadas sobre el estado del reservorio desaparece progresivamente con el tiempo. Esta propiedad asegura que el estado actual del sistema dependa principalmente de las entradas recientes, lo cual es crucial para la estabilidad y capacidad de generalización de la red. El cumplimiento de la ESP depende principalmente del radio espectral de la matriz de pesos del reservorio, es decir, del valor máximo del módulo de sus autovalores. Para que se mantenga la estabilidad del sistema, este valor debe ser, en general, menor que uno [14].

2.2.1.2 Entrenamiento y parámetros

El entrenamiento de estas ESN es diferente a las RNN ya que como hemos mencionado anteriormente únicamente se entrena la capa de salida, lo cual elimina la necesidad de aplicar retropropagación a través del tiempo como en las RNN tradicionales [14].

El entrenamiento consiste en ajustar los pesos de salida utilizando los estados internos del reservorio como entrada a una regresión supervisada. Esto permite obtener la salida deseada sin necesidad de entrenar las conexiones internas, lo que reduce significativamente el coste computacional [14]

Los parámetros clave de estas redes son los siguientes:

- ***N_x:*** Tamaño del reservorio
- ***ρ (W):*** Radio espectral (debe ser <1)
- ***c:*** Tasa de conexión
- **α:** Escala de pesos de entrada
- ***l:*** Tasa de fuga
- ***f (X):*** Función de activación (*tanh*)

2.2.1.3 Aplicaciones en Series Temporales

Las ESN se suelen utilizar en tareas que implican datos secuenciales como predicción, clasificación o detección de anomalías en series temporales [16]. En el ámbito bursátil, destacan por su capacidad para adaptarse a la no linealidad y la alta volatilidad del mercado. Se han empleado con éxito para predecir precios de acciones, índices e incluso criptomonedas, mostrando un rendimiento competitivo frente a modelos como LSTM cuando se configuran correctamente [16].

2.2.1.4 Ventajas y limitaciones

Las ESN ofrecen varias virtudes frente a otras arquitecturas: su eficiencia computacional supera a la de RNN y LSTM tradicionales, eliminan la necesidad de una retropropagación compleja, capturan patrones temporales con mínima parametrización y mantienen una buena capacidad de generalización incluso con conjuntos de datos reducidos. Sin embargo, también presentan ciertas restricciones: dependen en gran medida de una correcta configuración del reservorio, su rendimiento puede deteriorarse si los hiperparámetros no se afinan adecuadamente, resultan sensibles a entradas mal normalizadas y carecen de la memoria explícita de las LSTM, lo que les dificulta adaptarse a secuencias muy largas sin ajustes adicionales [17].

Por tanto, las ESN se presentan como una alternativa eficiente para tareas de predicción de series temporales con recursos limitados, siempre que se realice una adecuada selección de parámetros.

2.2.2 *Kolmogorov-Arnold Network* (KAN)

2.2.2.1 Arquitectura y funcionamiento

Las *Kolmogorov-Arnold Network* (KAN) son unas redes neuronales basadas en el teorema de representación de Kolmogorov-Arnold [18]. Este teorema expresa que cualquier función multivariable continua puede expresarse como suma de funciones univariantes continuas. Estas redes surgen debido a que las redes tradicionales pueden ser cajas negras, mientras que KAN busca aportar interpretabilidad estructural [19].

A diferencia de arquitecturas tradicionales como las MLPs (Perceptrones Multicapa, red neuronal con varias capas densas) o las CNNs (Redes Neuronales Convolucionales), que utilizan funciones de activación fijas como ReLU o sigmoide para transformar las entradas, las KAN prescinden de estas funciones estándar [19]. Esto se sustituye ya que los nodos aprenden directamente de funciones univariantes. Cada nodo de la red ajusta funciones específicas a sus entradas y después se combinan con pesos [19]. Esta estructura se representa en la Figura 4, donde se observa cómo cada nodo transforma sus entradas mediante funciones univariantes aprendidas, en lugar de aplicar activaciones fijas como en redes tradicionales.

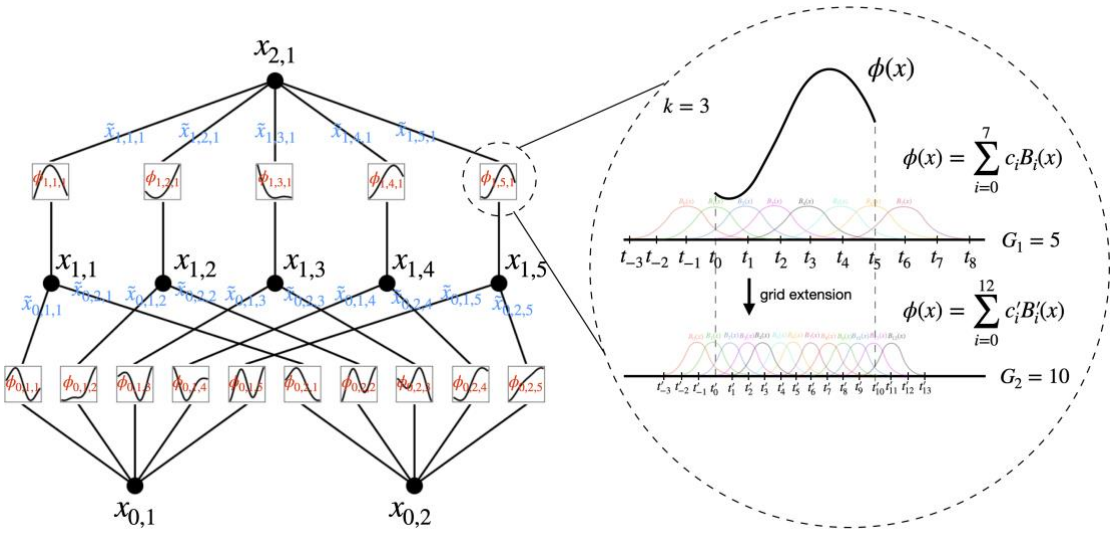

Figura 4. Estructura interna de una Kolmogorov-Arnold Network (KAN) [20].

2.2.2.2 Entrenamiento y parámetros

El entrenamiento de las *Kolmogorov-Arnold Networks* (KAN) es bastante diferente al de redes neuronales más clásicas como las MLP o LSTM. En lugar de ajustar pesos entre capas con funciones de activación fijas, aquí el objetivo es aprender directamente funciones univariantes que transforman cada variable de entrada de forma individual.

Estas funciones se suelen representar mediante *splines* (funciones formadas por segmentos polinómicos suaves que se conectan entre sí garantizando continuidad), polinomios u otras formas parametrizadas [19], y como son diferenciables, permiten usar algoritmos de optimización estándar como descenso por gradiente o Adam. En este proceso no solo se ajustan los pesos de salida, sino también los parámetros internos de cada función, lo que da al modelo una mayor flexibilidad.

La Figura 5 ilustra este proceso de entrenamiento, mostrando cómo cada nodo aplica una transformación univariante parametrizada sobre sus entradas, antes de combinarlas mediante pesos entrenables.

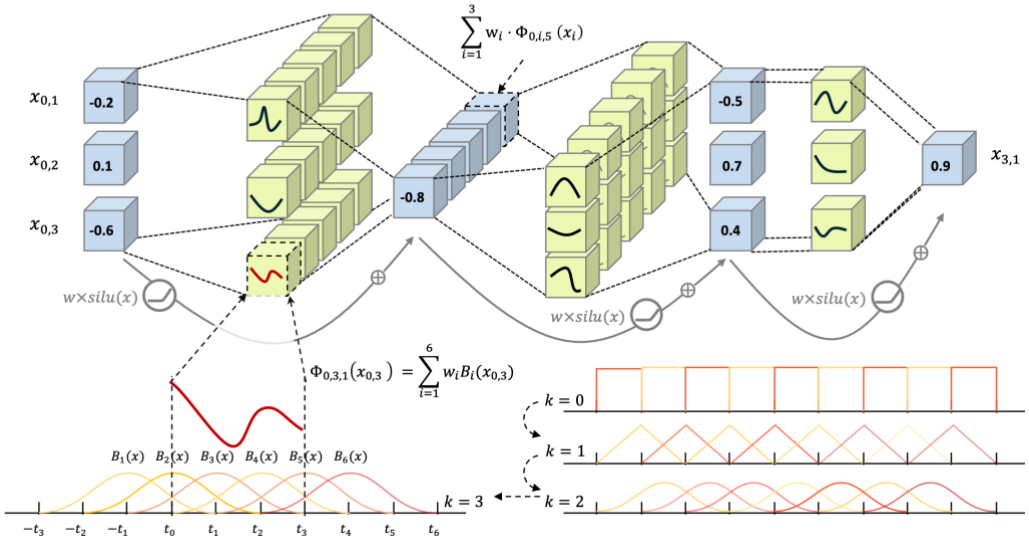

Figura 5. Funcionamiento del entrenamiento en una KAN [21].

A diferencia de otras redes, las KAN no necesitan funciones de activación predefinidas, lo que las hace más adaptables, aunque también más exigentes a la hora de entrenar. Cada nodo tiene que aprender una transformación útil sobre su variable de entrada, y si no se aplica una buena regularización, el modelo puede acabar sobre ajustando fácilmente.

Los principales hiperparámetros que afectan al rendimiento y estabilidad del modelo son:

- ***k:*** Número de funciones por nodo: determina la capacidad expresiva del modelo.
- ***d:*** Profundidad de la red: controla cuántas composiciones de funciones se aplican.
- ***λ:*** Regularización tanto sobre los pesos como sobre la forma de las funciones (por ejemplo, suavidad de los *splines*), para evitar que el modelo se ajuste demasiado a los datos de entrenamiento.
- ***b:*** Tipo de función base: elección entre *splines*, funciones polinómicas u otras parametrizaciones.

Entrenar una KAN requiere más ajustes y planificación que otros modelos, pero a cambio ofrece una representación mucho más explicable de las relaciones entre variables. Esto resulta especialmente útil en tareas financieras, donde además de buenos resultados se necesita saber cómo y por qué se han tomado ciertas decisiones.

2.2.2.3 Aplicaciones en las series temporales

Las KAN son una arquitectura reciente dentro del campo de las redes neuronales, por lo que su aplicación en entornos financieros aún está en desarrollo. A pesar de esto hay experimentos recientes en predicción financiera, análisis climático y otras tareas de modelado temporal [19].

Su característica más destacable para su uso en series temporales es su buena capacidad para generalizar aún con menos datos en mercados con alta volatilidad o baja disponibilidad. Sin embargo, para las que las KAN han destacado más hasta el momento han sido la predicción de cotizaciones de acciones y criptomonedas y la estimación de funciones de valor o políticas en entornos tipo trading automatizado [19].

Aunque aún en fase de investigación, su potencial para tareas de predicción financiera está empezando a ser explorado con resultados prometedores.

2.2.2.4 Ventajas y limitaciones

Entre sus principales ventajas destacan la interpretabilidad ya que cada transformación univariante puede visualizarse y analizarse, la eficiencia computacional frente a modelos de redes recurrentes y la buena generalización incluso con conjuntos de datos reducidos. Por otro lado, presentan retos: la sintonía de hiperparámetros resulta más crítica que en arquitecturas tradicionales, la documentación y herramientas específicas son todavía escasas, y sin una regularización adecuada pueden incurrir en sobreajuste.

2.2.3 Long Short-Term Memory Networks (LSTM)

2.2.3.1 Arquitectura y funcionamiento

Las *Long Short-Term Memory Networks* (LSTM) son una variante mejorada de las redes neuronales recurrentes de las cuales ya hemos hablado antes. Estas redes mejoran sustancialmente la capacidad de las RNN tradicionales en el aspecto de resolver el problema del desvanecimiento y la explosión del gradiente que dificultan el aprendizaje de dependencias a largo plazo en secuencias temporales [22]. La clave para que su arquitectura contenga esta mejora es la inclusión de una estructura llamada celda de memoria, que mantiene la información durante periodos prolongados.

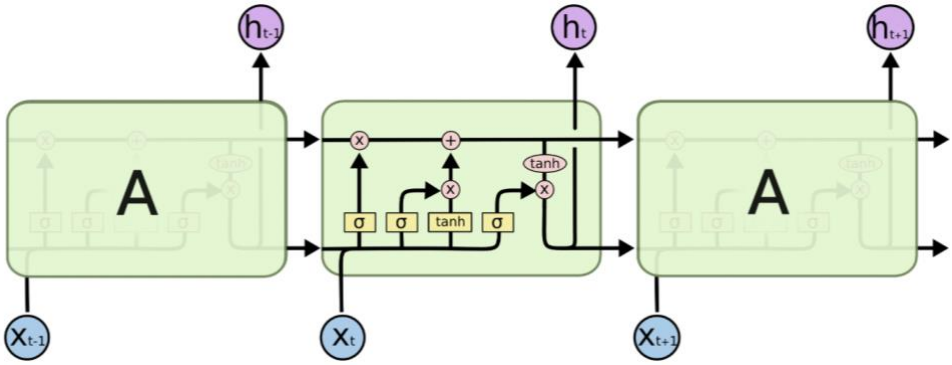

Figura 6. Estructura interna de una celda LSTM [23].

Tal y como se muestra en la Figura 6, una celda LSTM contiene tres compuertas fundamentales: la compuerta de olvido, que decide qué información se descarta del estado de la celda; la compuerta de entrada, que determina qué nueva información se almacena; y la compuerta de salida, que controla qué parte de la celda influye en la salida del modelo.

Estas compuertas se controlan mediante funciones sigmoides y permiten que el modelo decida qué información guardar y cuál eliminar en cada paso de tiempo. Gracias a este mecanismo las LSTM son capaces de capturar los patrones tanto a corto como a largo plazo en datos secuenciales [22]. Estas compuertas hacen que el modelo retenga mejor la información relevante lo convierte en una opción muy sólida para la predicción de series temporales con alta complejidad estructural [24].

2.2.3.2 Entrenamiento y parámetros

El entrenamiento de una LSTM sigue los mismos principios que una red neuronal recurrente utilizando retropropagación para ajustar los pesos a lo largo de la secuencia. Sin embargo, las compuertas de memoria evitan la probabilidad de desvanecimiento debido a que permiten que el gradiente se propague fácilmente.

Los principales parámetros para considerar durante el entrenamiento son:

- ***u:*** Número de unidades LSTM por capa.
- ***l:*** Profundidad o el número de capas.
- ***p:*** *Dropout*.
- ***f:*** Función de activación (tahn y sigmoide).

Además, se suelen usar optimizadores como *Adam*, que ofrecen buenos resultados en entornos con gradientes ruidosos y secuencias largas [25].

2.2.3.3 Aplicaciones en series temporales

Las LSTM son altamente eficaces en la predicción de series temporales debido a su gran capacidad de identificar patrones a largo plazo. Se utilizan mayoritariamente en ámbitos como la predicción bursátil y de criptomonedas [26]. Son muy utilizadas debido a que su capacidad para modelar la no linealidad y la alta variabilidad del mercado [26]. Además de que son capaces de manejar secuencias largas, aspecto clave para los mercados bursátiles.

2.2.3.4 Ventajas y limitaciones

Las principales virtudes de las LSTM son su capacidad para aprender dependencias a largo plazo, la estabilidad del entrenamiento frente a RNN tradicionales y su amplia disponibilidad en frameworks como TensorFlow y PyTorch. Sin embargo, presentan un coste computacional elevado y requieren un ajuste fino de hiperparámetros para evitar el sobreajuste; además, su lógica interna, aunque más estructurada que la de una RNN simple, sigue siendo relativamente opaca cuando se explica a terceros [27].

2.2.4 *Liquid Neural Network* (LNN)

2.2.4.1 Arquitectura y funcionamiento

Las *Liquid Neural Networks* (LNN) son un tipo de red neuronal que introduce una dinámica no lineal en su arquitectura interna, lo cual les permite adaptarse mejor a datos en tiempo real y a secuencias con alta variabilidad temporal. A diferencia de otras redes como las LSTM o las ESN, las LNN se basan en ecuaciones diferenciales que modelan el comportamiento de sus neuronas, lo que hace que su estado evolucione de forma continua en el tiempo [28].

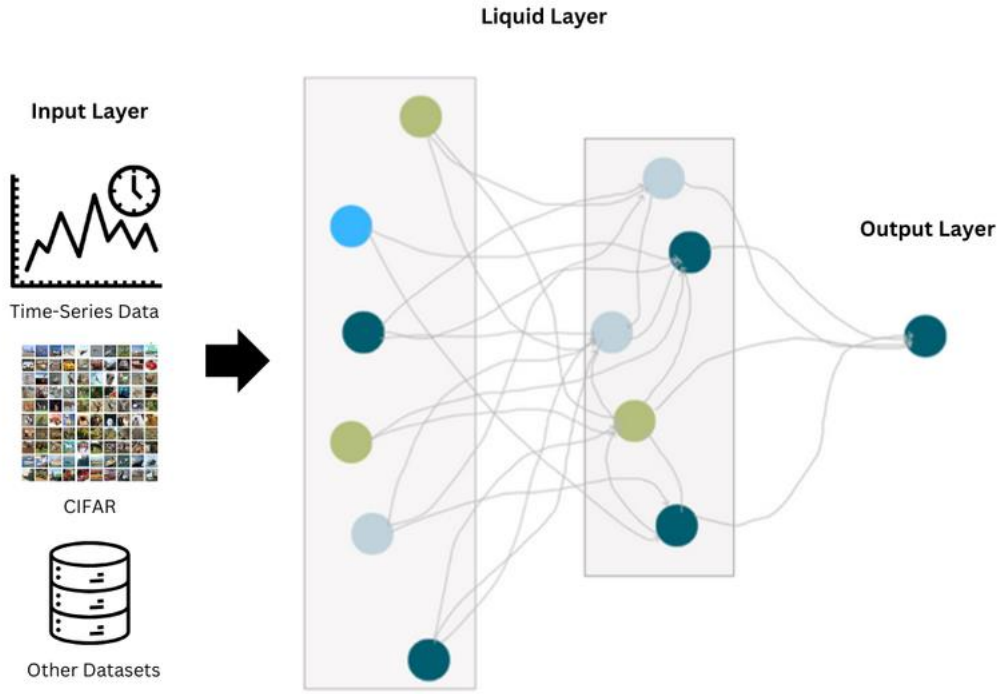

Figura 7. Representación simplificada del flujo de información en una Liquid Neural Network [29].

Estas redes no trabajan con capas fijas, sino que tienen un núcleo recurrente cuya dinámica está controlada por sistemas diferenciales. Como se observa en la Figura 7, la información fluye desde la capa de entrada hacia una capa líquida con dinámica adaptable, y finalmente se dirige a la capa de salida. La salida de cada neurona no depende simplemente de activaciones clásicas, sino de la solución de una ecuación diferencial que describe cómo cambia el estado interno con el tiempo. Esto les da una capacidad de memoria temporal fluida y adaptable, algo muy útil cuando las señales de entrada varían mucho o tienen estructuras no constantes.

Una de las ideas clave detrás de estas redes es que, en lugar de añadir más capas o más neuronas para capturar comportamientos complejos, se aumenta la expresividad del modelo incorporando una dinámica más rica en cada neurona. Esta propiedad hace que las LNN tengan una mejor capacidad de generalización, incluso con menos datos o con menor tamaño de red [30].

2.2.4.2 Entrenamiento y parámetros

El entrenamiento de las LNN no sigue exactamente la misma lógica que otras redes. Aunque se puede seguir utilizando retropropagación del error, al estar basadas en sistemas dinámicos no lineales, se requiere calcular derivadas de funciones que evolucionan en el tiempo, lo cual se realiza mediante técnicas como el método de *Euler* o *Runge-Kutta* para resolver las ecuaciones diferenciales involucradas [31]. En la Figura 8 se muestra un ejemplo esquemático de una LNN aplicada al procesamiento de señales visuales, donde puede observarse la interacción entre diferentes tipos de neuronas dentro de la capa líquida.

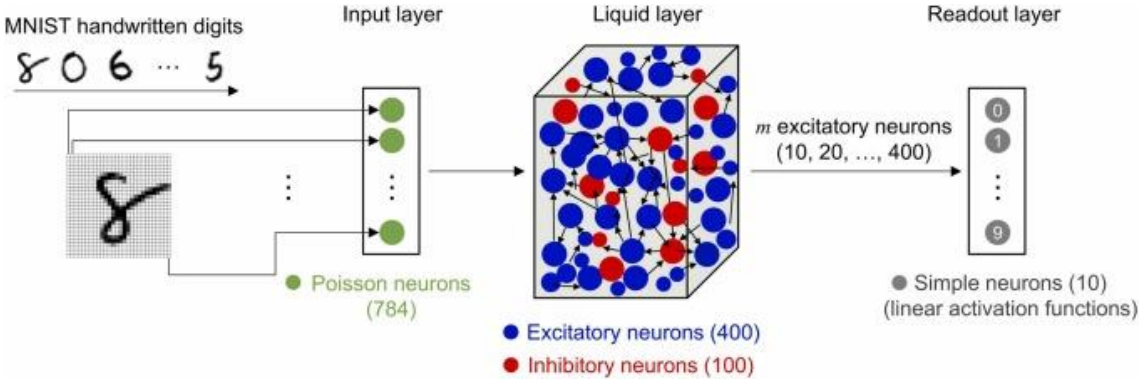

Figura 8. Ejemplo visual de una Liquid Neural Network con neuronas excitatorias e inhibitorias conectadas dinámicamente [32].

Algunos de los parámetros clave en estas redes son:

- ***g:*** Los coeficientes del sistema dinámico de cada neurona (a veces llamados parámetros del "liquido").
- ***i:*** Los parámetros que definen las condiciones iniciales de las neuronas.
- ***s:*** El tamaño del paso temporal en la integración numérica.
- ***h***: Hidden Size es la dimensión del estado líquido [28].

Estas redes son más sensibles al ruido en los datos y al ajuste fino de los hiperparámetros, pero si se entrenan correctamente, ofrecen resultados muy competitivos, especialmente en tareas que requieren una alta capacidad de adaptación a patrones temporales complejos.

2.2.4.3 Aplicaciones a series temporales

Las *Liquid Neural Networks* se han utilizado principalmente en campos como la robótica autónoma, predicción de señales biológicas y procesamiento de datos sensoriales en tiempo real. Sin embargo, recientemente se ha empezado a investigar su aplicación en tareas financieras y de predicción de series temporales con resultados prometedores [33].

En concreto, su capacidad para adaptarse dinámicamente a entradas cambiantes las hace interesantes para mercados bursátiles, donde los datos varían rápidamente y no siempre siguen un patrón constante. También han mostrado buen rendimiento en tareas como detección de anomalías y modelado de series multivariantes.

2.2.4.4 Ventajas y desventajas

Las Liquid Neural Networks destacan por su alta capacidad de adaptación a secuencias con dinámicas complejas y por su fuerte generalización fuerte incluso con una menor cantidad de datos. También destaca que requieren una menor necesidad de profundidad de red gracias a la complejidad interna de las neuronas. Además, proporcionan un buen rendimiento en entornos donde los datos cambian en tiempo real.

Sin embargo, el uso de ecuaciones diferenciales conlleva un coste computacional elevado y exige conocimientos especializados para su correcta implementación y ajuste. La poca madurez de las bibliotecas y herramientas específicas para LNN complica su uso en entornos de producción. Por último, presentan una mayor sensibilidad a hiperparámetros como el paso temporal o los coeficientes del sistema dinámico [28].

2.3 Fundamentos técnicos del entrenamiento de modelos de redes neuronales

2.3.1 Métricas de evaluación

Para comparar el rendimiento de las distintas redes neuronales que se estudiarán en este trabajo, no es suficiente con saber si predicen mucho o poco. Hace falta un sistema claro y cuantificable que permita medir cómo de buenas son esas predicciones, tanto en el conjunto de entrenamiento como en el de prueba. Por eso, se van a utilizar varias métricas estándar que evalúan distintos aspectos del error, la precisión y la capacidad de generalización del modelo.

En primer lugar, el Error Cuadrático Medio (*Mean Squared Error, MSE*) calcula la media de las diferencias al cuadrado entre el valor real y el predicho. Penaliza más los errores grandes, por lo que es útil cuando interesa castigar desviaciones importantes. Cuanto más bajo sea, mejor el modelo [34]. Matemáticamente se expresa como:

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

Derivado de esta métrica, el Error Cuadrático Medio de la Raíz (*Root Mean Squared Error, RMSE*): simplemente toma la raíz cuadrada del *MSE*. Se expresa en las mismas unidades que los datos originales, lo que hace que sea más fácil de interpretar a nivel práctico. Un RMSE bajo indica que el modelo ajusta bien los datos sin errores grandes [34]. Su interpretación es:

$$\mathrm{RMSE} = \sqrt{\mathrm{MSE}}$$

Por otro lado, el Error Absoluto Medio (*Mean Absolute Error, MAE*) calcula la media del valor absoluto de los errores. A diferencia del MSE, no penaliza tanto los errores grandes, lo que la convierte en una métrica más robusta frente a valores atípicos (*outliers*). El MAE proporciona una visión clara del error promedio en términos absolutos [34]:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

El Coeficiente de Determinación (R^2) indica qué proporción de la variabilidad de los datos es explicada por el modelo. Su valor va de 0 a 1 (aunque puede ser negativo si el modelo es peor que una media simple), y cuanto más cerca de 1 esté, mejor será el ajuste [34]. Se calcula como:

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

En contextos donde se requiere una perspectiva relativa del error, se utilizan métricas porcentuales como el Error Porcentual Absoluto Medio (*Mean Absolute Percentage Error, MAPE*). Esta métrica mide cuánto se desvía la predicción en términos relativos respecto al valor real. Por ejemplo, un MAPE del 5% indica que, en media, el modelo falla un 5% respecto al valor real. No obstante, tiene como inconveniente que puede ser inestable cuando los valores reales se acercan a cero [34]. Es expresado como porcentaje:

$$\mathrm{MAPE} = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

Para mitigar estos problemas, se recurre al Error Porcentual Absoluto Medio Simétrico (*Symmetric Mean Absolute Percentage Error, SMAPE*), una variante del MAPE que soluciona algunos de sus problemas, sobre todo cuando los valores reales son muy bajos. Lo hace al promediar el valor absoluto del error sobre la

media de los valores real y predicho. Esto da una visión más equilibrada del error porcentual [35]:

$$\text{SMAPE} = \frac{100}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{(|y_i| + |\hat{y}_i|)/2}$$

2.3.2 Función de coste: Error cuadrático (MSE)

Para el entrenamiento de redes neuronales, es fundamental utilizar una función para medir el error que comete el modelo al realizar las predicciones. En este caso se usa el parámetro MSE (Mean Squared Error). Como ya se explicó brevemente en la sección anterior, el MSE calcula la media del cuadrado de las diferencias entre los valores reales y_i y los valores predichos $\hat{y}_i$. Matemáticamente se expresa como [36]:

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

La principal característica de esta fórmula es que penaliza de forma más severa los errores más grandes debido al uso del cuadrado, lo cual es muy útil en contextos financieros donde una predicción desviada puede tener un mayor impacto que un número de errores pequeños. En la práctica esta se implementa con bibliotecas como *skicit-learn* [37], que nos permite calcular el MSE de forma directa mediante la función *sklearn.metrics.mean_squared_error,* que evalúa el rendimiento de modelos con arrays de predicciones y valores reales.

2.3.3 Algoritmo de optimización Adam

En el entrenamiento de las redes neuronales, los algoritmos de optimización son los encargados de minimizar la función de coste, ajustando progresivamente los pesos del modelo. Uno de los más utilizados actualmente es *Adam (Adaptive Moment Estimation),* debido a su eficiencia computacional y a sus buenos resultados en escenarios complejos y ruidosos [25].

Adam combina dos técnicas: el uso del momento (el optimizador Momentum) y también la varianza adaptativa (RMSProp). El momento hace que el optimizador no se base únicamente en el gradiente actual, sino también en los gradientes anteriores, suavizando las actualizaciones. RMSProp, por su parte, ajusta la tasa de aprendizaje para cada peso en función de su variabilidad. En cada iteración calcula dos valores, el primer momento, que es la media móvil del gradiente, y el segundo momento, que es la media móvil del cuadrado del gradiente. Esto le permite ajustar la tasa de aprendizaje de cada peso automáticamente y corregir el sesgo inicial [38].

Adam es muy útil en el contexto de las series temporales financieras, donde hay ruido o las funciones de pérdida son complejas. Además, converge más rápido que otros métodos anteriormente utilizados como el descenso por gradiente estocástico (SDG) y requiere menor ajuste de hiperparámetros [38].

Los principales hiperparámetros de Adam son:

- **lr** (learning rate): controla el tamaño de los pasos que da el momento en cada actualización. Si es demasiado alto puede volverse inestable y si es demasiado bajo puede no converger nunca.
- **β1**: controla cuanto influye el primer momento
- **β2**: controla cuanto influye el segundo momento.
- **ε**: valor pequeño para evitar divisiones por cero.

2.3.4 Escalado de variables y normalización

Los modelos de redes neuronales son muy sensibles a la escala de los datos. Si las variables tienen precios muy distintos se dificulta mucho el aprendizaje del modelo. Para ello utilizamos el escalado de variables, una técnica de

preprocesamiento que ajusta los datos para que se encuentren en un rango común. Sin un escalado adecuado, las variables con rangos más amplios pueden dominar el aprendizaje.

En cuanto a técnicas de escalado existen dos principales. La primera es la normalización, que consiste en la transformación de los datos para que se encuentren en un rango común, generalmente [0, 1]. La segunda es la estandarización que ajusta los datos para que tengan una media de 0 y una desviación estándar de 1.

Estas técnicas se implementan mediante escaladores como *MinMaxScaler* (normalización) y *StandardScaler* (estandarización), ambos disponibles en bibliotecas como *Scikit-learn* [39]. En el contexto de las series temporales financieras es habitual utilizar ambos enfoques, aunque el uso de MinMaxScaler puede ser preferible cuando se trabaja con precios, ya que estos suelen moverse en rangos positivos acotados y no siguen una distribución normal. Por otra parte, StandardScaler puede utilizarse en indicadores más estacionarios o centrados y es más sensible a valores extremos [40].

Otra opción sería el uso de RobustScaler, que se basa en la mediana y en el rango intercuartílico, lo que lo hace más resistente a valores atípicos. No obstante, se suele utilizar para series con presencia clara de valores extremos, cosa que no ocurre en las series temporales financieras [40].

2.3.5 Creación de secuencias: Ventana deslizante

El entrenamiento de una red neuronal no es posible si se le presentan una lista de datos sin contexto. Para que el modelo aprenda patrones temporales los datos necesitan contexto temporal.

La técnica más utilizada es una ventana deslizante, una técnica que convierte una serie temporal en un conjunto de pares de entrada/salida. Cada entrada sería una secuencia de N pasos anteriores y cada salida el precio siguiente a predecir.

Esta ventana se desliza paso a paso sobre la serie generando ejemplos como:

Entrada: $X\_t\text{-}N, \ldots, X\_t\text{-}1 \rightarrow$ Salida: X_t.

Este enfoque permite a la red capturar al modelo patrones locales y dependencias temporales [41]. En el caso de las series temporales financieras, esta técnica es especialmente útil para aprender dinámicas de precios que dependen del historial reciente de comportamiento del activo.

2.3.6 Entrenamiento por lotes (batches)

Las redes neuronales no se entrenan con la totalidad de los datos al mismo tiempo, como hemos mencionado en el apartado anterior. En su lugar, los datos se dividen en lotes llamados *batches*. Cada batch contiene un número fijo de ejemplos que se procesan juntos en cada iteración.

Esta estrategia aporta múltiples beneficios al entrenamiento de los modelos. Por un lado, mejora la eficiencia computacional, al reducir la necesidad de memoria y favorecer procesos paralelos. Además, los pesos se actualizan con mayor frecuencia tras cada lote procesado, lo que optimiza la dinámica de aprendizaje. Finalmente, esta estrategia ayuda a obtener una mejor generalización del modelo gracias a la introducción de cierta aleatoriedad en los datos utilizados durante el entrenamiento [41].

En el contexto de las series temporales cada batch contiene secuencias contiene secuencias generadas mediante un *window_size* fijo. Si, por ejemplo, se utiliza un batch_size de 32, esto significa que en cada iteración se procesan 32

secuencias distintas al mismo tiempo, y cada secuencia representa una ventana de tamaño fijo.

El tamaño de este batch afecta al entrenamiento de forma que valores pequeños generalizan mejor, pero pueden generar más ruido en la actualización de los pesos, mientras que valores grandes suavizan el proceso; pero pueden llevar a un peor rendimiento en test [41], [42].

Por tanto, elegir correctamente el tamaño del batch es clave para el equilibrio entre estabilidad del entrenamiento y capacidad de generalización.

2.3.7 Tasa de aprendizaje y programación dinámica

La tasa de aprendizaje (*learning rate, lr*), es un parámetro fundamental en el entrenamiento de redes neuronales, ya que determina el tamaño de los pasos que da el modelo al actualizar los pesos. Una tasa de aprendizaje alta puede hacer que el modelo no converja o que oscile alrededor del mínimo, mientras que si es muy baja puede hacer que el entrenamiento sea extremadamente lento o que el modelo se quede atrapado en mínimos locales [43].

Aunque en Adam ya se mencionó la tasa de aprendizaje como uno de sus hiperparámetros, es importante destacar que a veces no es conveniente mantener la misma durante todo el entrenamiento. Para ello se utilizan técnicas de programación dinámica de la tasa de aprendizaje (*learning rate scheduling*) que hacen que esta se ajuste en función del progreso del entrenamiento [43], [44].

Una de las técnicas más utilizadas es *ReduceLROnPlateau*, disponible en bibliotecas como *Pytorch*. Este scheduler reduce la tasa de aprendizaje cuando una de las métricas monitoreadas ha dejado de mejorar durante un número determinado de épocas. Los principales parámetros de *ReduceLROnPlateau* son [44]:

- Mode: Determina si se busca minimizar o maximizar la métrica monitoreada.
- Factor: Factor por el cual se reducirá la tasa de aprendizaje.
- Patience: Número de épocas con estancamiento antes de reducir la tasa.
- Threshold: Cambio mínimo en la métrica para considerarse una mejora significativa.

Por ejemplo, si se establece patience=10 y factor=0.1, la tasa de aprendizaje se reducirá en un 90% si la métrica no mejora en 10 épocas consecutivas. Esta técnica es especialmente útil en el entrenamiento de modelos sobre series temporales financieras, donde el comportamiento de la pérdida puede estancarse debido a la naturaleza ruidosa y no estacionaria de los datos [43].

2.3.8 Propagación del error y ajuste de pesos

El entrenamiento de una red neuronal se basa en dos fases principales: la propagación hacia adelante (*forward pass*) y la propagación hacia atrás (*backpropagation*), seguidas por el ajuste de pesos mediante un algoritmo de optimización [45].

En este proceso es clave el uso del gradiente, una magnitud que indica la dirección y magnitud del cambio de la función de pérdida respecto a cada peso. El gradiente permite saber cómo modificar cada parámetro del modelo para reducir el error, y es la base sobre la que se apoyan los algoritmos de optimización como el descenso por gradiente [46].

En la fase de la propagación hacia adelante los datos se transmiten a través de la red capa por capa. Cada neurona calcula una combinación lineal de sus entradas, aplica una función de activación y pasa el resultado a la siguiente capa. Este proceso se hace hasta conseguir la salida final del modelo.

El error se calcula mediante la comparación de la salida predicha y la salida real utilizando alguna función de pérdida como podría ser el error medio cuadrático (*MSE*).

La propagación hacia atrás se hace una vez calculado el error. Este se propaga hacia atrás para determinar cómo cada peso contribuyó a ese error. Utilizando la regla de la cadena del cálculo diferencial, se computan los gradientes de la función de pérdida respecto a cada peso de la red [43]. Este proceso eficiente permite actualizar los pesos de manera que minimicen el error en futuras iteraciones.

Finalmente, en el ajuste de pesos se actualizan los pesos de la red utilizando algún algoritmo de optimización como el descenso por gradiente o el optimizador Adam. Estos algoritmos ajustan los pesos de forma que se reduce la función de pérdida, haciendo que la red aprenda patrones relevantes en los datos.

Este ciclo se repite durante múltiples épocas, lo que permite que el modelo mejore progresivamente su rendimiento en la tarea específica.

2.3.9 Indicadores técnicos en series temporales financieras: el MACD

El Moving Average Convergence Divergence (MACD) es un oscilador de momento que compara dos medias móviles exponenciales (EMAs) para describir la fuerza y la dirección de la tendencia de un activo. Su formulación clásica es

$$\mathrm{MACD}_t = \mathrm{EMA}_{12}(P)_t - \mathrm{EMA}_{26}(P)_t$$

$$\mathrm{Signal}_t = \mathrm{EMA}_9(\mathrm{MACD})_t$$

$$\mathrm{Hist}_t = \mathrm{MACD}_t - \mathrm{Signal}_t$$

En ese esquema, “12-26-9” se refiere a los periodos de cada EMA: la EMA de 12 días, la EMA de 26 días y luego una EMA de 9 días sobre el MACD. Esos valores (12, 26 y 9) se han convertido en estándar porque la EMA de 12 días reacciona rápido a cambios recientes, la de 26 días es más suave y la de 9 días permite filtrar el ruido en la serie MACD. Cuando la línea MACD cruza por encima de la línea Signal (EMA de 9), el histograma (MACD − Signal) pasa de negativo a positivo, señalando un posible giro alcista; si cruza por debajo, ocurre lo contrario, indicando posible giro bajista [47].

En la Figura X se muestra un ejemplo real para Apple (AAPL): en el panel superior aparece la cotización diaria con sus EMAs de 12 y 26 días; en el panel inferior, la línea MACD (EMA 12 − EMA 26), la línea Signal (EMA 9 del MACD) y el histograma (MACD − Signal).

Figura 9. Indicadores técnicos en series temporales financieras: el MACD [48].

2.3.9.1 Capacidad de Generalización, overfitting y regularización

En aprendizaje supervisado, la capacidad de generalización se mide por el rendimiento del modelo en datos no vistos, es decir, si el error en validación crece mientras el de entrenamiento sigue disminuyendo, hablamos de overfitting, es decir, el modelo está memorizando ruido del conjunto de ajuste en lugar de extraer patrones transferibles [43]. Para evitarlo sin añadir términos de penalización en la función de pérdida, se recurre a la regularización mediante dropout, que aleatoriamente inactiva neuronas durante el entrenamiento obligando a la red a no depender de rutas fijas [43], y al early stopping, que interrumpe el proceso cuando la métrica de validación deja de mejorar tras varias épocas y restaura automáticamente los pesos correspondientes al mejor punto validado [49].

Adicionalmente, emplear una división cronológica de los datos en entrenamiento, validación sobre el entrenamiento y prueba garantiza que ningún dato futuro influya en el ajuste de hiperparámetros [50],

2.4 Comparativa y retos en la predicción con redes neuronales

A lo largo del apartado anterior hemos descrito en detalle cuatro arquitecturas distintas: ESN, KAN, LSTM y LNN. Cada una presenta características propias, ventajas concretas y ciertas limitaciones, pero para poder compararlas de forma justa y rigurosa es fundamental establecer unas métricas comunes. En este apartado veremos cómo se pueden evaluar y comparar estos modelos en contextos reales, y qué retos aparecen cuando se aplican a series temporales financieras.

2.4.1 Ventajas y Limitaciones de cada Modelo

En la tabla 1 se resumen las principales fortalezas y debilidades de cada una de las arquitecturas analizadas:

Modelo	Ventajas	Limitaciones
ESN	Entrenamiento rápido, buena generalización con pocos datos, eficiencia computacional.	Requiere ajustar bien el reservorio, puede ser sensible a la normalización.
KAN	Muy interpretables, compactas, buena generalización con pocos datos.	Difíciles de entrenar, alta sensibilidad a los hiperparámetros, menos documentación.
LSTM	Excelente en capturar dependencias a largo plazo, robustas con secuencias largas.	Coste computacional alto, entrenamiento lento, caja negra en su funcionamiento.
LNN	Adaptación dinámica, muy buena con señales no lineales y en tiempo real.	Complejidad matemática, entrenamiento delicado, poco soporte en *frameworks*.

Tabla 1. Principales fortalezas y limitaciones de las arquitecturas ESN, KAN, LSTM y LNN.

A partir de esta comparativa general se puede ver que cada arquitectura tiene sus propios puntos fuertes, pero también ciertas limitaciones que hay que tener en cuenta. No existe un modelo universal que funcione mejor en todos los escenarios, y por eso es clave entender bien el contexto del problema. Si se dispone de pocos datos o recursos limitados, puede interesar una ESN o una KAN bien configurada. En cambio, si se prioriza precisión en series largas y complejas, LSTM o LNN pueden rendir mejor, aunque a cambio de un coste computacional más alto y mayor complejidad en el entrenamiento.

Esta diversidad de fortalezas y debilidades es precisamente lo que justifica llevar a cabo una evaluación comparativa con datos reales en condiciones controladas, que es lo que se abordará en los siguientes capítulos del proyecto.

3 Diseño experimental

Este capítulo describe el proceso completo seguido para diseñar, implementar y evaluar los modelos de predicción de series temporales bursátiles. Se parte de un conjunto de datos históricos reales y se aplican distintos métodos de preprocesamiento, modelado y entrenamiento. El objetivo es asegurar que la comparación entre arquitecturas se realiza en condiciones controladas, objetivas y reproducibles.

Primero se describen las características de los datasets utilizados y las variables seleccionadas. Luego se explican las transformaciones y técnicas de preprocesamiento que se han aplicado para preparar los datos correctamente. También se detalla cómo se ha implementado cada uno de los modelos (ESN, KAN, LSTM y LNN), y cómo se ha configurado el entrenamiento de forma unificada para que todos trabajen bajo los mismos criterios. Por último, se indican las herramientas, librerías y el entorno de desarrollo usados para llevar a cabo los experimentos.

De esta forma, se establece la base técnica necesaria para interpretar adecuadamente los resultados del capítulo siguiente, garantizando la trazabilidad y la transparencia del proceso.

3.1 Metodología de trabajo

3.1.1 Descripción del dataset

Para el desarrollo de este trabajo se han utilizado datos diarios de precios de acciones de las diez compañías con mayor porcentaje en el índice S&P 500. Esta selección permite cubrir empresas altamente representativas del mercado estadounidense, con alta liquidez y una amplia base de inversores, lo que facilita la robustez del análisis. Las empresas seleccionadas han sido: Apple (AAPL), Microsoft (MSFT), Amazon (AMZN), NVIDIA (NVDA), Meta Platforms (META), Tesla (TSLA), Alphabet Class A (GOOGL), Broadcom Inc. (AVGO), Berkshire Hathaway (BRK) y Eli Lilly and Company (LLY) [51].

Los datos, obtenidos de forma gratuita a través de Yahoo Finance [52], incluyen para cada compañía la fecha de registro; los precios de apertura (Open), máximo (High), mínimo (Low) y cierre (Close); el precio de cierre ajustado (AdjClose); y el volumen total negociado en la sesión (Volume). Entre estas variables, se ha seleccionado AdjClose como variable objetivo (target) para la predicción.

La cobertura temporal se ha estructurado en tres períodos: en primer lugar, se consideran los datos correspondientes a los últimos cinco años; a continuación, se amplía la ventana a los últimos diez años; y, por último, se emplea el rango histórico máximo disponible para las diez compañías, tomando como fecha de inicio el 18 de mayo de 2012 (día en que Meta Platforms salió a bolsa), lo que equivale a aproximadamente trece años de registros.

Estas tres versiones del dataset permiten evaluar cómo afecta la longitud de la serie histórica al rendimiento de los modelos, y analizar si una mayor cantidad de datos mejora necesariamente la capacidad de predicción.

3.1.2 Preprocesamiento aplicado

El preprocesamiento de los datos es una etapa crucial en el desarrollo de modelos de redes neuronales, especialmente en el ámbito financiero, donde la calidad y adecuación de los datos pueden influir significativamente en el rendimiento del modelo. A continuación, se detallan las transformaciones aplicadas:

Ordenación cronológica

Se ordenaron los datos de forma ascendente según la columna *Date* para preservar la secuencia temporal y evitar la introducción de información futura en el entrenamiento del modelo. Esta práctica es fundamental en el análisis de series temporales, donde el orden de los datos afecta directamente a la modelización y predicción.

Definición de variable objetivo

La variable objetivo (AdjClose_next) se definió como el precio de cierre ajustado del día siguiente, aplicando un desplazamiento hacia atrás (AdjClose.shift(-1)). Este enfoque permite al modelo anticipar el valor del activo en la sesión posterior a partir de la información disponible hasta el día actual.

Se eligió la variable AdjClose en lugar de Close como objetivo de predicción, ya que el precio ajustado incorpora los efectos de dividendos y divisiones de acciones (splits), proporcionando así una representación más precisa del valor real del activo para los inversores. Esta elección permite un análisis más fiel de la rentabilidad histórica, evitando distorsiones por eventos corporativos, y se encuentra respaldada en estudios aplicados a predicción bursátil [53].

Selección de variables predictoras

En este trabajo, las variables de entrada combinan información de precios tradicionales, un indicador técnico y la rentabilidad diaria. En primer lugar, se emplearon los precios de apertura (Open), máximo (High), mínimo (Low), cierre (Close) y el volumen negociado (Volume). A continuación, se añadieron tres columnas derivadas del indicador MACD (Moving Average Convergence Divergence), calculado con parámetros (14, 21, 9):

- MACD_14_21_9: diferencia entre las medias exponenciales de 14 y 21 días.
- MACDs_14_21_9: media exponencial de 9 días de la línea MACD.
- MACDh_14_21_9: diferencia entre la línea MACD y la línea de señal.

Este indicador técnico es comúnmente utilizado para detectar cambios en la dirección de la tendencia y ha demostrado ser útil como input en modelos de machine learning aplicados a datos bursátiles [47].

Por último, se incluyó la variable DailyReturn, definida como

$$\text{DailyReturn}_t = \frac{\text{AdjClose}_t}{\text{AdjClose}_{t\text{-}1}} - 1$$

con el fin de capturar la variabilidad relativa del activo y su dinámica más reciente. De este modo, para predecir AdjClose_{t+1} el modelo solo conoce AdjClose_{t} y AdjClose_{t-1}. Esto permite capturar la variabilidad relativa del activo hasta el día actual sin filtrar información del día siguiente.

Normalización

Se utilizó el método MinMaxScaler para escalar las variables de entrada y la variable objetivo transformada al rango [0, 1]. Esta normalización asegura que todas las variables contribuyan equitativamente al entrenamiento del modelo y permite una convergencia más rápida y estable. Diferentes estudios han demostrado que la elección de una técnica de escalado adecuada puede influir significativamente en el rendimiento de modelos de clasificación y regresión [54].

3.1.3 Implementación de los modelos

En este apartado se presentan las implementaciones en PyTorch de los modelos introducidos en el capítulo 2: ESN, KAN, LSTM y LNN. Cada una de ellas ha sido construida respetando fielmente las formulaciones teóricas descritas en la literatura original, con el objetivo de mantener la coherencia entre la estructura

propuesta y su traducción a código. Como punto de partida, se consultaron diversos repositorios de GitHub que sirvieron de referencia en el diseño de las arquitecturas.

3.1.3.1 Echo State Network (ESN)

La arquitectura ESN implementada en este trabajo reproduce de forma directa el modelo de reservoir computing descrito en el apartado 2.2.1. La red ha sido desarrollada en PyTorch y se compone de una capa de entrada conectada a un reservorio interno, cuyas conexiones se inicializan aleatoriamente y permanecen fijas durante todo el entrenamiento. La única capa entrenada es la de salida, aplicada mediante regresión supervisada sobre los estados del reservorio. Esta configuración sigue las recomendaciones originales de Lukosevicius [14], y se ha respetado sin introducir ajustes adicionales. Aunque todo el código se ha escrito desde cero, la estructura del reservorio y el flujo de datos se han inspirado en la librería PyTorch-ESN de Stefan Onardo [55].

Para la implementación del ESN se definieron varios hiperparámetros fundamentales que equilibran capacidad de modelado y coste computacional. El tamaño del reservorio se ha establecido en N_x = 2000 neuronas, un tamaño suficiente para captar relaciones temporales complejas sin penalizar en exceso el rendimiento. El radio espectral se ha fijado en $\rho(W)$ = 0.99 para asegurar el cumplimiento de la *Echo State Property*, permitiendo al modelo mantener una dinámica estable sin perder memoria de entradas pasadas. Después se ha establecido la tasa de conexión en c = 0.1 (input_scaling), lo que implica que solo un 10% de las conexiones del reservorio están activas. Esta configuración busca un equilibrio entre diversidad dinámica y eficiencia computacional.

En cuanto a la escala de pesos de entrada se ha fijado en α = 0.1. Este valor controla la amplitud de las señales que entran al reservorio, y se ha mantenido bajo para evitar la saturación de las unidades internas. Por otra parte, la tasa de fuga se ha fijado en l = 0.5 (connectivity). Este parámetro permite controlar la inercia temporal de los estados internos, y un valor intermedio permite adaptarse a patrones de corto y medio plazo. Finalmente se ha optado por una función de activación $f(x)$ = tanh, en línea con la literatura sobre redes de tipo reservoir [14].

3.1.3.2 Kolmogorov-Arnold Network (KAN)

La configuración empleada para la KAN se define mediante cuatro hiperparámetros globales que mantienen el espíritu de Wang y Uhler [19] y simplifican su implementación. Aunque la implementación se ha desarrollado íntegramente para este trabajo, la organización de las funciones univariantes y la optimización de la ejecución se inspiraron en el repositorio Blealtan/efficient-kan de GitHub [56].

En primer lugar, el número de funciones univariantes por nodo k =1 establece una única transformación lineal (peso y sesgo) por variable de entrada, en lugar de múltiples splines diferenciables. A continuación, el tipo de base univariante b sustituye las splines originales por funciones afines simples, sin knots (estos son los puntos de unión entre segmentos polinómicos en una spline) ni términos polinomiales, reduciendo así la complejidad de la parametrización. La profundidad de la red d = 1 implica que la arquitectura consta únicamente de una capa de transformaciones univariantes seguida de la capa lineal de salida, tal como en la propuesta base. Por último, el coeficiente de regularización λ = 0 refleja la decisión de no aplicar penalizaciones explícitas ni sobre las funciones univariantes ni sobre los pesos de la capa final, permitiendo evaluar el comportamiento de la KAN en su forma más pura y directa.

3.1.3.3 Long Short-Term Memory (LSTM)

La arquitectura LSTM implementada en este trabajo reproduce de forma directa el modelo descrito en el apartado 2.2.3. La red ha sido desarrollada en PyTorch y se compone de una única capa LSTM seguida de una capa densa que genera la predicción final. Esta configuración incorpora celdas de memoria con compuertas de entrada, olvido y salida, lo que permite al modelo retener información relevante a largo plazo y adaptarse a secuencias temporales complejas. Se ha respetado la formulación clásica sin añadir mecanismos adicionales. Como punto de partida, la definición de la capa LSTM y el manejo de los estados ocultos se inspiró en los ejemplos del repositorio oficial pytorch/examples de PyTorch [57].

Para la configuración experimental se establecieron los siguientes hiperparámetros: el número de unidades ocultas se fijó en $\boldsymbol{u}$ = 128, lo que proporciona una capacidad adecuada para capturar patrones temporales sin incurrir en un coste excesivo; la profundidad de la red se limitó a una única capa LSTM $\boldsymbol{l}$ = 1, suficiente para modelar dependencias en los datos sin sobreajuste; la tasa de dropout se estableció en $\boldsymbol{p}$ = 0,2, introduciendo regularización entre capas y ayudando a prevenir el sobreajuste; y las funciones de activación clásicas se mantuvieron conforme a la implementación por defecto de PyTorch, esto es, sigmoides en las compuertas de entrada, olvido y salida, y tangente hiperbólica (tanh) para generar la información candidata y escalar el estado de la celda [58].

3.1.3.4 Liquid Neural Network (LNN)

La arquitectura LNN implementada en este trabajo reproduce de forma directa el modelo descrito en el apartado 2.2.4. La red ha sido desarrollada en PyTorch y se compone de una celda líquida con dinámica no lineal controlada por una constante de tiempo entrenable, seguida de una capa densa que genera la predicción final. El estado interno de la red evoluciona de forma continua mediante una combinación ponderada entre el estado anterior y una transformación no lineal de la entrada actual. Esta dinámica se ajusta automáticamente durante el entrenamiento, y permite al modelo adaptarse a patrones temporales complejos sin necesidad de una arquitectura profunda. Se ha respetado la formulación teórica sin incorporar modificaciones adicionales. Como punto de partida, el diseño de la celda de constante de tiempo y la dinámica discreta se inspiraron en el repositorio raminmh/liquid_time_constant_networks de GitHub [59].

Los hiperparámetros establecidos para el correcto funcionamiento de esta red neuronal se han fijado según estudios anteriores. Para los coeficientes del sistema dinámico $\boldsymbol{g}$ se han inicializado los pesos sinápticos W_x W_h utilizando la estrategia xavier_uniform_, siguiendo las recomendaciones de Hasani et al. [28], para garantizar una dinámica estable y evitar problemas de gradientes desvanecientes o explosivos.

En cuanto a las condiciones iniciales $\boldsymbol{i}$ el estado oculto inicial se ha establecido en cero al comienzo de cada secuencia, siguiendo la práctica recomendada en Hasani et al. [28], donde se justifica esta decisión como una forma de preservar la neutralidad inicial del sistema y facilitar una evolución dinámica no sesgada de la red. Esta elección es especialmente adecuada en tareas de predicción con secuencias independientes, como las series temporales bursátiles utilizadas en este trabajo.

El Paso temporal de integración $\boldsymbol{s}$ en esta implementación no se ha utilizado un paso temporal explícito, ya que no se integra una ecuación diferencial mediante métodos como Euler o Runge-Kutta. En su lugar, se ha empleado una actualización discreta directa del estado interno $\boldsymbol{h}$, inspirada en la dinámica de

las LNN, donde el parámetro α actúa como constante de mezcla. Esta decisión permite simplificar el modelo sin comprometer su capacidad de adaptación temporal, siguiendo enfoques prácticos similares a los utilizados en implementaciones discretas de LNN [28].

Finalmente, el tamaño de la capa líquida se ha fijado en $h = 128$ neuronas. Este valor proporciona un compromiso entre capacidad para capturar relaciones temporales complejas y eficiencia en tiempo de entrenamiento y memoria.

3.1.4 Configuración del entrenamiento

Para garantizar una comparación justa entre los modelos ESN, KAN, LSTM y LNN, se ha seguido una configuración de entrenamiento unificada, con los mismos criterios de preprocesamiento, tamaño de secuencia, división de datos y condiciones de aprendizaje.

Creación de secuencias temporales

Se ha utilizado una ventana deslizante de tamaño 60 para generar las secuencias de entrada, lo que permite capturar relaciones temporales de corto y medio plazo. Esta elección se basa en el trabajo de Fischer y Krauss [60], donde se emplea dicha longitud de ventana en tareas de predicción bursátil con redes LSTM. Aunque su estudio se centra en arquitecturas recurrentes, este valor se ha extendido al resto de modelos (ESN, KAN, LNN) con el fin de mantener la coherencia y asegurar una comparación homogénea.

División de los datos

Para preservar la estructura temporal y evitar fugas de información futura, es decir, que el modelo llegue a ver durante el entrenamiento datos de fechas posteriores a las que está aprendiendo y obtenga así resultados irreales, los datos se han dividido de forma secuencial. Primero se separó el conjunto de pruebas (test) manteniendo el orden cronológico, y posteriormente se reservó una parte del entrenamiento como validación. Este enfoque sigue las recomendaciones de Hyndman y Athanasopoulos [4], quienes destacan que, en series temporales, es fundamental respetar la dependencia temporal para no distorsionar los resultados.

También se dividieron los datos de forma secuencial en tres subconjuntos. En primer lugar, se asignó un 75% de las observaciones para entrenamiento y un 25% para test, respetando el orden cronológico de las muestras. Posteriormente, se reservó un 20% del conjunto de entrenamiento como validación. Esta estrategia permite evaluar el rendimiento del modelo en datos no vistos sin romper la continuidad temporal, y sigue el valor por defecto de la función *train_test_split* de Scikit-learn, utilizado como referencia en este trabajo [50].

Configuración de los lotes (batches)

Todos los modelos (ESN, KAN, LSTM y LNN) se entrenaron con batch_size = 32. El uso de lotes pequeños se basa en el estudio de Masters y Luschi [41], que señala 32 como tamaño de equilibrio entre estabilidad y capacidad de generalización en redes LSTM; se adopta el mismo valor en las demás arquitecturas para mantener condiciones comparables.

Durante la fase de entrenamiento se activó shuffle=True para mezclar aleatoriamente las ventanas generadas (sin alterar el orden interno de cada secuencia). Al introducir este grado de aleatoriedad entre lotes, se reduce la alta autocorrelación entre ejemplos consecutivos y se consigue un aprendizaje más robusto, ya que el modelo no ve siempre patrones temporales muy próximos en cada mini-batch.

En cambio, para validación y prueba se estableció shuffle=False, de modo que los datos se procesan estrictamente en orden cronológico. Mantener la secuencia temporal intacta en estos conjuntos evita que el modelo "vea" ejemplos de fechas posteriores a las que está evaluando: si diéramos la vuelta o mezcláramos esas ventanas, podríamos estar comparando predicciones generadas con información futura dentro del mismo lote, lo cual produciría estimaciones irreales del rendimiento y, en última instancia, fugas de información futura.

Optimizador y parámetros

Todos los modelos fueron entrenados utilizando el optimizador Adam, por su capacidad para adaptarse dinámicamente al gradiente de cada parámetro y su eficacia en problemas con ruido o datos no estacionarios [25]. Se ha mantenido la configuración estándar de Adam en PyTorch:

- $\beta_1 = 0.9$,
- $\beta_2 = 0.999$,
- $\varepsilon = 1e\text{-}8$,

La única modificación realizada ha sido el ajuste de la tasa de aprendizaje (learning_rate), que se ha fijado en 0.001, valor comúnmente utilizado en tareas de regresión sobre series temporales y recomendado en múltiples estudios [43].

Todos los modelos se entrenaron con Adam, por su adaptación automática del paso de aprendizaje y robustez frente a datos ruidosos [25]. Se mantuvieron los valores estándar del algoritmo y se fijó learning rate = 0.001, valor recomendado en el trabajo original de Kingma y Ba [25].

Scheduler de aprendizaje

Se empleó ReduceLROnPlateau, que disminuye la tasa de aprendizaje cuando la métrica de validación se estanca. La configuración utilizada fue mode='min', factor=0.5, patience=10, threshold=1e-4, verbose=True, siguiendo los valores de referencia de la documentación oficial de PyTorch [44].

Bucle de entrenamiento y Early Stopping

El número máximo de épocas se fijó en 200 para el entrenamiento. En cada época, el modelo entrena con los batches de entrenamiento y luego se evalúa en el conjunto de validación. Seguidamente, se guarda la versión del modelo que logre la menor pérdida de validación. Si en 15 épocas consecutivas la pérdida de validación no mejora, se detiene anticipadamente el entrenamiento y se restauran los pesos del mejor modelo. Este criterio de paciencia (15 épocas) es una práctica de implementación común basada en estudios de sobreajuste, si bien no proviene de un mandato bibliográfico estricto [49].

3.1.5 Evaluación del rendimiento y métricas

Para la evaluación de los modelos se utilizan las métricas definidas en la sección 2.3.1: MSE, RMSE, MAE, R^2, MAPE y SMAPE[34].

Todas estas métricas se calculan tanto sobre el conjunto de validación (para ajustar hiperparámetros y decidir el Early Stopping) como sobre el conjunto de prueba (para la comparación final de los cuatro modelos).

En la práctica, empleamos funciones de la biblioteca *scikit-learn* para las métricas estándar (mean_squared_error, mean_absolute_error, r2_score, mean_absolute_percentage_error), y una implementación propia para SMAPE asegurando así la reproducibilidad de los resultados.

Para completar la fase metodológica se establecen dos elementos adicionales que guiarán la interpretación de los resultados en el Capítulo 4.

Gráfica de precio real vs. predicción

Se hizo una representación del precio real y el precio inferido por los modelos en el conjunto de pruebas para demostrar su capacidad de aprendizaje.

Curvas de pérdida (train / validación)

Tras cada época se registran las pérdidas de entrenamiento y de validación. La representación gráfica de ambas curvas permite comprobar la convergencia del proceso de optimización y detectar signos de sobreajuste cuando la pérdida de validación deja de acompañar a la de entrenamiento. Esta práctica sigue las recomendaciones de Hyndman & Athanasopoulos sobre la importancia de la inspección visual del error en modelos de series temporales [4].

Coste de entrenamiento y tamaño del modelo

Por último, se incorpora una segunda tabla que detalla el coste computacional de cada red. En ella se consignan el número total de épocas, la época en la que la validación alcanzó su mínimo, el tiempo total de entrenamiento, el tiempo transcurrido hasta esa mejor época y el número de parámetros ajustables. Estos datos permiten valorar la eficiencia temporal y la ligereza de cada arquitectura, de modo que la primera tabla informa sobre la calidad predictiva y la segunda sobre los recursos necesarios para obtenerla.

3.2 Herramientas

3.2.1 Entorno de desarrollo

Todo el proyecto se ha desarrollado en Python 3.10 dentro del IDE PyCharm Professional (2024.1), aprovechando su integración con sistemas de control de versiones Git y su soporte nativo para entornos virtuales. Para las fases exploratorias y de análisis de datos se han empleado Jupyter Notebooks, que facilitan la combinación de código ejecutable y visualizaciones.

El entorno de ejecución de los notebooks y del script principal es un entorno virtual creado con venv, que incluye las librerías necesarias sin interferir con la instalación global de Python. Para el entrenamiento de los modelos se ha utilizado PyTorch 2.0, ejecutándose sobre CPU o, cuando está disponible, MPS (Metal Performance Shaders) en macOS para acelerar el cálculo en GPU de Apple.

3.2.2 Librerías y estructura del código

El código fuente está organizado de forma modular en la carpeta src/, separando claramente la lógica de los modelos, el preprocesamiento y las utilidades. Para el entrenamiento y análisis se emplean Jupyter Notebooks, mientras que los archivos de código independientes (por ejemplo, model_esn.py, model_kan.py, model_lstm.py, model_lnn.py) contienen la definición y carga de cada arquitectura. En estos notebooks y módulos se utilizan PyTorch (torch, torch.nn, torch.optim) para definir las arquitecturas, las funciones de pérdida (MSELoss), el optimizador Adam y el scheduler ReduceLROnPlateau; NumPy y pandas para las operaciones numéricas y la manipulación de datos en formato tabular; scikit-learn para aplicar MinMaxScaler al escalado de entradas y objetivos y para calcular métricas como MSE, MAE y R^2; joblib para serializar los objetos de escalado (scaler_X.pkl, scaler_y.pkl) y asegurar la reproducibilidad; y matplotlib para generar las gráficas comparativas (precio real frente a predicción y frente a baseline

4 Análisis de resultados

Se aplicaron los modelos (ESN, KAN, LSTM y LNN) a las series temporales de precios ajustados de diez de las compañías de mayor capitalización del S&P500. Para cada activo se ha llevado a cabo un análisis en tres horizontes (5 años, 10 años y el máximo histórico) con el fin de valorar cómo varía la capacidad predictiva de cada modelo según la longitud de la serie.

En cada sección dedicada a un horizonte temporal se integran de manera articulada las métricas cuantitativas (MSE, RMSE, MAE, R^2, MAPE y SMAPE), la comparación gráfica entre precio real y predicción, la evolución de la función de pérdida durante entrenamiento y validación, así como la información sobre los recursos computacionales empleados (tiempo de entrenamiento, número de épocas hasta convergencia y tamaño del modelo). Este enfoque permite apreciar de forma homogénea la relación entre precisión y eficiencia de cada red, sentando las bases para la discusión comparativa final y la identificación del modelo más apropiado en función de diferentes escenarios temporales y de complejidad de datos.

4.1 Apple (AAPL)

4.1.1 Horizonte de 5 Años

4.1.1.1 Resumen de métricas

Modelo	MSE	RMSE	MAE	R^2	MAPE (%)	SMAPE (%)
ESN	129.67	11.39	9.78	0.71	4.29%	4.41%
KAN	31.55	5.62	4.53	0.93	2.06%	2.08%
LSTM	89.83	9.48	8.02	0.80	3.55%	3.63%
LNN	348.41	18.67	16.39	0.22	7.15%	7.49%

Tabla 2. Resumen de métricas de modelos para Apple - Horizonte de 5 años.

Como se observa en la tabla 2, la KAN obtiene el mejor equilibrio precisión-eficiencia con un MAPE de 2,06 % y R^2=0,93, superando claramente a ESN (4,29 % y 0,71) y LSTM (3,55 % y 0,80), mientras que la LNN muestra el mayor error (7,15 %) y menor capacidad explicativa (R^2=0,22).

4.1.1.2 Precio real vs. Predicción

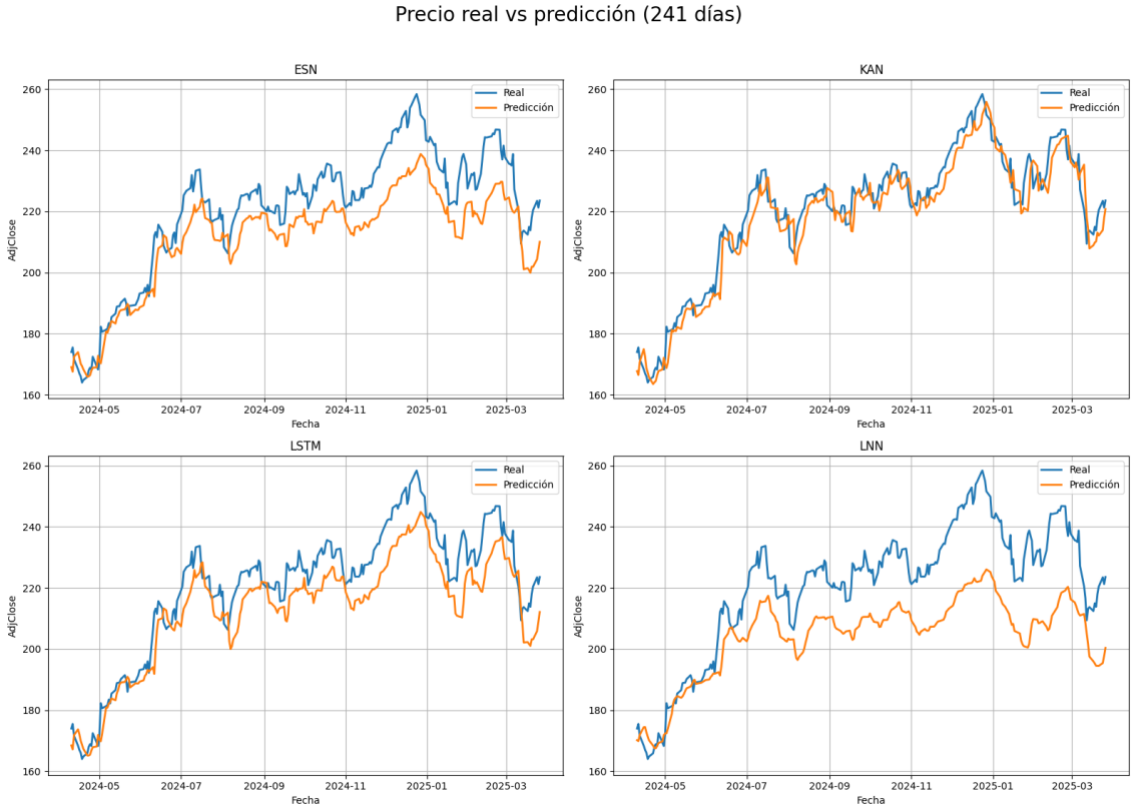

Figura 10. Precio real vs. predicción de los modelos para Apple - Horizonte de 5 años.

Como se observa en la Figura 10, el gráfico la KAN sigue con fidelidad los picos de la serie histórica, ESN y LSTM presentan un ligero desfase alcista y la LNN tiende a subestimar los extremos.

4.1.1.3 Curvas de pérdida

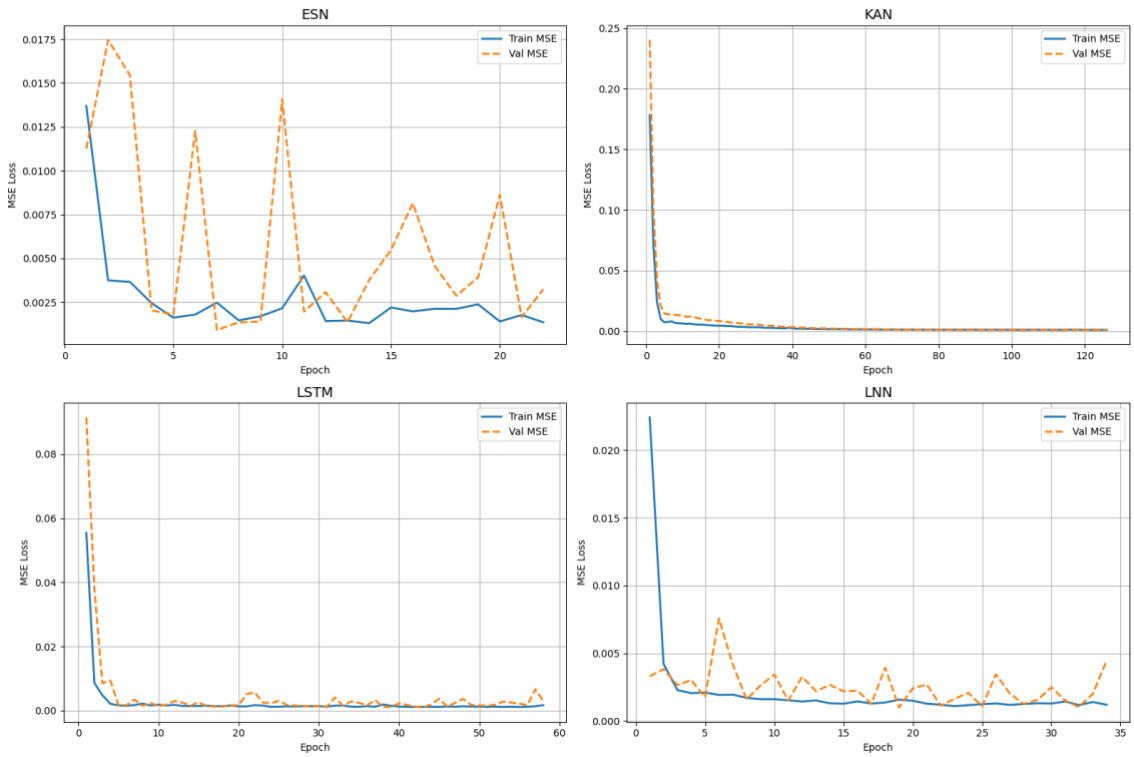

Figura 11. Curvas de pérdida de entrenamiento y validación de los modelos para Apple - Horizonte de 5 años.

Según el gráfico de la Figura 11 todas las redes convergen en pocas épocas, pero la validación de ESN muestra oscilaciones esporádicas en MSE (épocas 2, 10 y 20), mientras que KAN y LSTM tienen descenso suave y LNN exhibe algún repunte puntual en validación.

4.1.1.4 Coste de entrenamiento y tamaño del modelo

Modelo	Épocas	Mejor época	Tiempo de entrenamiento (s)	Tiempo a mejor época (s)	Mejor Val MSE	Parámetros
ESN	22	7	9.01s	3.31s	0.000919	2001
KAN	126	111	19.92s	17.51s	0.000959	28
LSTM	58	43	10.97s	8.37s	0.000976	71297
LNN	34	19	41.11s	23.31s	0.000999	17921

Tabla 3. Coste de entrenamiento y tamaño del modelo para Apple - Horizonte de 5 años.

Según los datos de la tabla 3, la KAN ofrece la mayor ligereza (28 parámetros) y converge en 17,5 s (época 111), frente a ESN (2 001 parámetros y 3,3 s), LSTM (71 297 parámetros y 8,4 s) y LNN (17 921 parámetros y 23,3 s), lo que refuerza su eficiencia.

4.1.2 Horizonte de 10 Años

4.1.2.1 Resumen de métricas

Modelo	MSE	RMSE	MAE	R^2	MAPE (%)	SMAPE (%)
ESN	276.20	16.62	13.20	0.69	6.27%	6.56%
KAN	21.08	4.59	3.56	0.98	1.83%	1.85%
LSTM	175.29	13.24	10.37	0.81	4.93%	5.12%
LNN	126.34	11.24	8.63	0.86	4.10%	4.23%

Tabla 4. Resumen de métricas de modelos para Apple - Horizonte de 10 años.

Tal y como aparece en la tabla 4, la KAN mantiene su ventaja con un MAPE de 1,83 % y R^2=0,98, mientras que ESN registra el mayor error (6,27 % y R^2=0,69). LSTM y LNN ofrecen un desempeño intermedio (MAPE 4,93 % y 4,10 %; R^2 0,81 y 0,86, respectivamente).

4.1.2.2 Precio real vs. Predicción

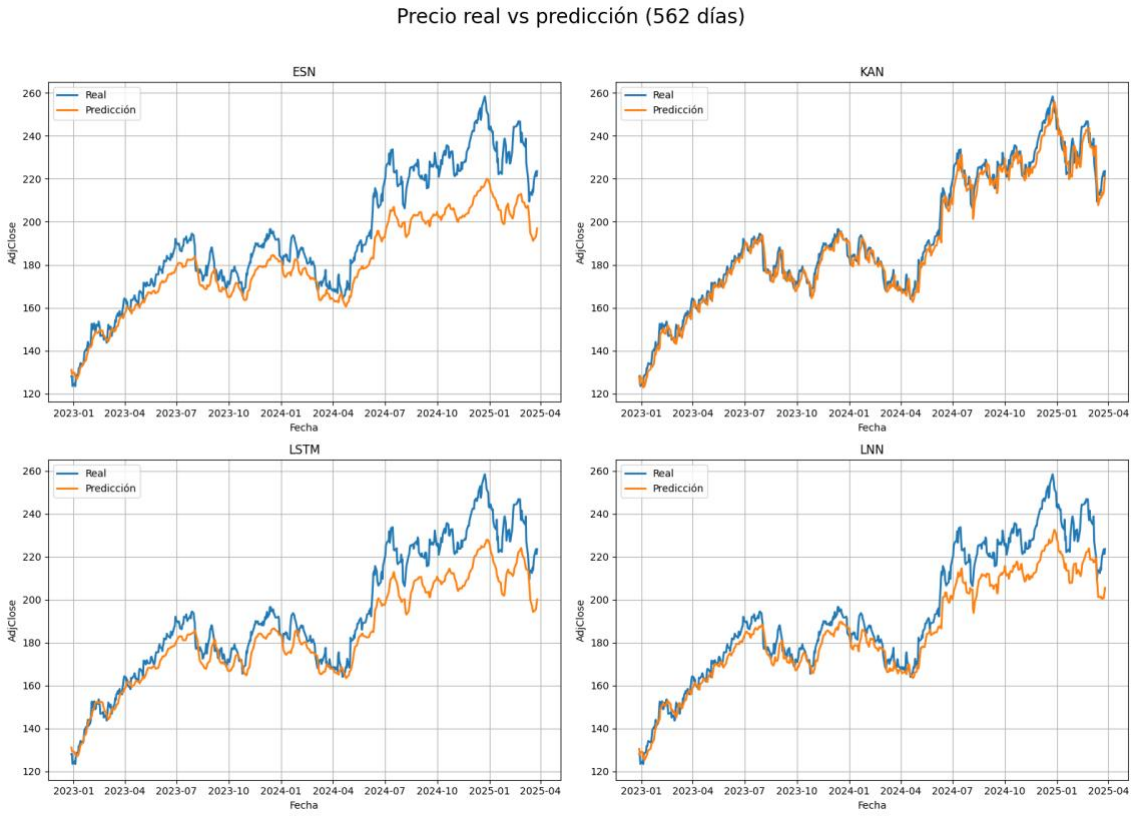

Figura 12. Precio real vs. predicción de los modelos para Apple - Horizonte de 10 años.

Como se observa en la figura 12, la predicción de la KAN se superpone casi por completo a la serie real, capturando fielmente los puntos altos y bajos. El ESN tiende a quedarse algo por debajo en los picos, y tanto LSTM como LNN suavizan levemente las oscilaciones más marcadas.

4.1.2.3 Curvas de pérdida

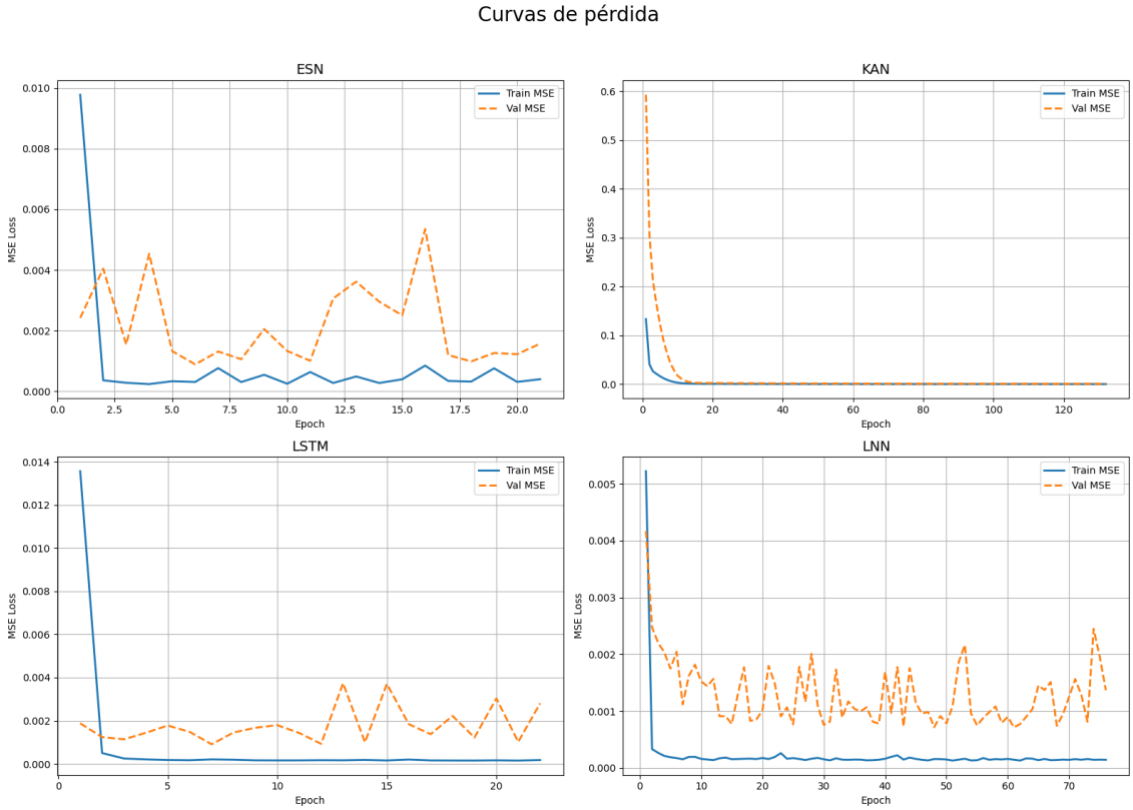

Figura 13. Curvas de pérdida de entrenamiento y validación de los modelos para Apple - Horizonte de 10 años.

Según los gráficos de la figura 13, KAN y LSTM presentan un descenso limpio y sin sobresaltos en entrenamiento y validación, lo que refleja un aprendizaje muy estable. Por su parte, ESN y LNN muestran algunas fluctuaciones en la pérdida de validación, señal de cierta inestabilidad puntual durante el ajuste.

4.1.2.4 Coste de entrenamiento y tamaño del modelo

Modelo	Épocas	Mejor época	Tiempo de entrenamiento (s)	Tiempo a mejor época (s)	Mejor Val MSE	Parámetros
ESN	21	6	16.74s	4.88s	0.000888	2001
KAN	132	117	43.95s	39.23s	0.000652	28
LSTM	22	7	10.49s	3.82s	0.000909	71297
LNN	76	61	185.55s	150.07s	0.000717	17921

Tabla 5. Coste de entrenamiento y tamaño del modelo para Apple - Horizonte de 10 años.

Según los datos de la tabla 5, la KAN vuelve a destacar por su ligereza: con solo 28 parámetros y convergiendo en menos de 45 s, logra un gran rendimiento. ESN y LSTM requieren miles de parámetros para un tiempo de ajuste más rápido por época, mientras que la LNN necesita varios minutos de entrenamiento para alcanzar su mejor validación.

4.1.3 Horizonte Máximo

4.1.3.1 Resumen de métricas

Modelo	MSE	RMSE	MAE	R^2	MAPE (%)	SMAPE (%)
ESN	647.45	25.45	20.65	0.41	10.50%	11.30%
KAN	22.85	4.78	3.72	0.98	2.07%	2.08%
LSTM	161.24	12.70	9.56	0.85	4.78%	4.95%
LNN	1795.30	42.37	36.68	-0.65	18.80%	21.15%

Tabla 6. Resumen de métricas de modelos para Apple - Horizonte máximo.

Como se observa en la tabla6, la KAN sigue destacando incluso con el histórico completo, con un MAPE de 2,07 % y R^2=0,98, manteniendo la precisión vista en horizontes más cortos. El ESN pierde fiabilidad (MAPE 10,50 %, R^2=0,41), el LSTM aguanta razonablemente bien (4,78 % y R^2=0,85) y la LNN se desdibuja con un error muy elevado (18,80 % y R^2 negativo).

4.1.3.2 Precio real vs. Predicción

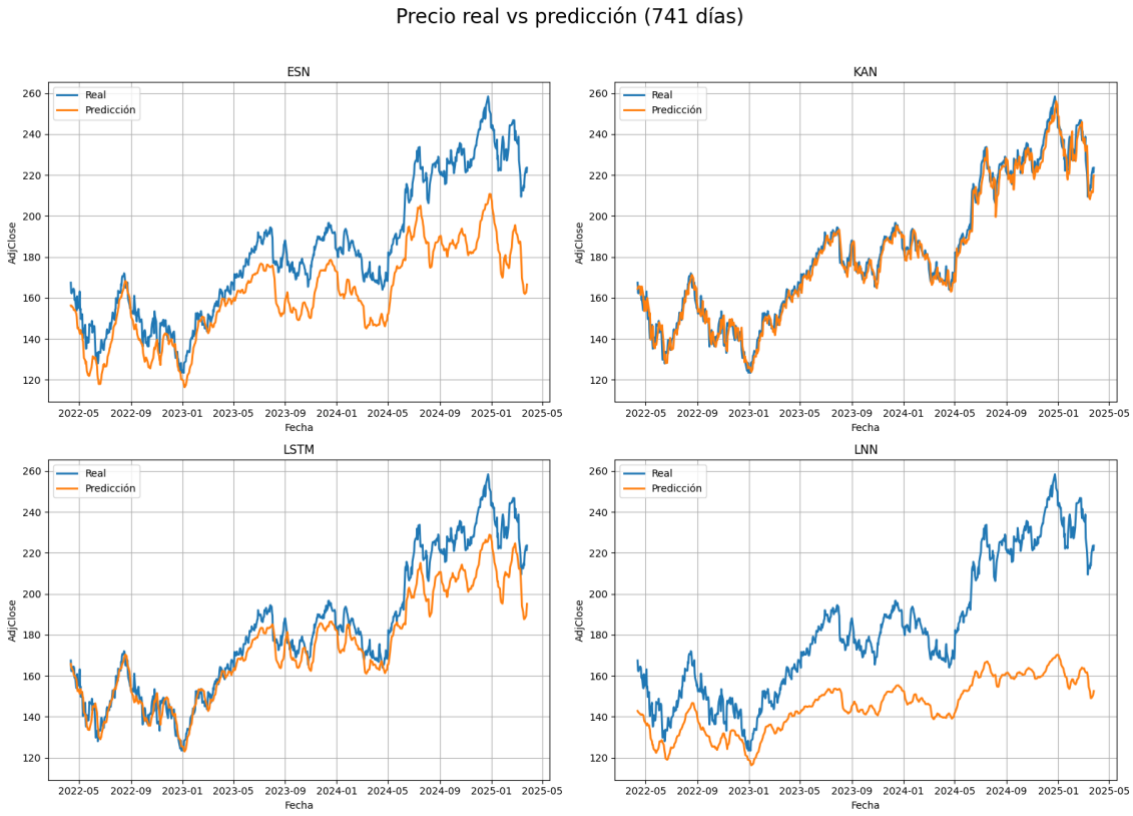

Figura 14. Precio real vs. predicción de los modelos para Apple - Horizonte máximo.

Tal y como queda reflejado en la figura 14, la KAN reproduce prácticamente todos los altibajos de la serie real. El LSTM sigue la tendencia general, pero suaviza picos y valles, el ESN tiende a subestimar en fases alcistas y la LNN presenta un ajuste demasiado conservador a lo largo de todo el periodo.

4.1.3.3 Curvas de pérdida

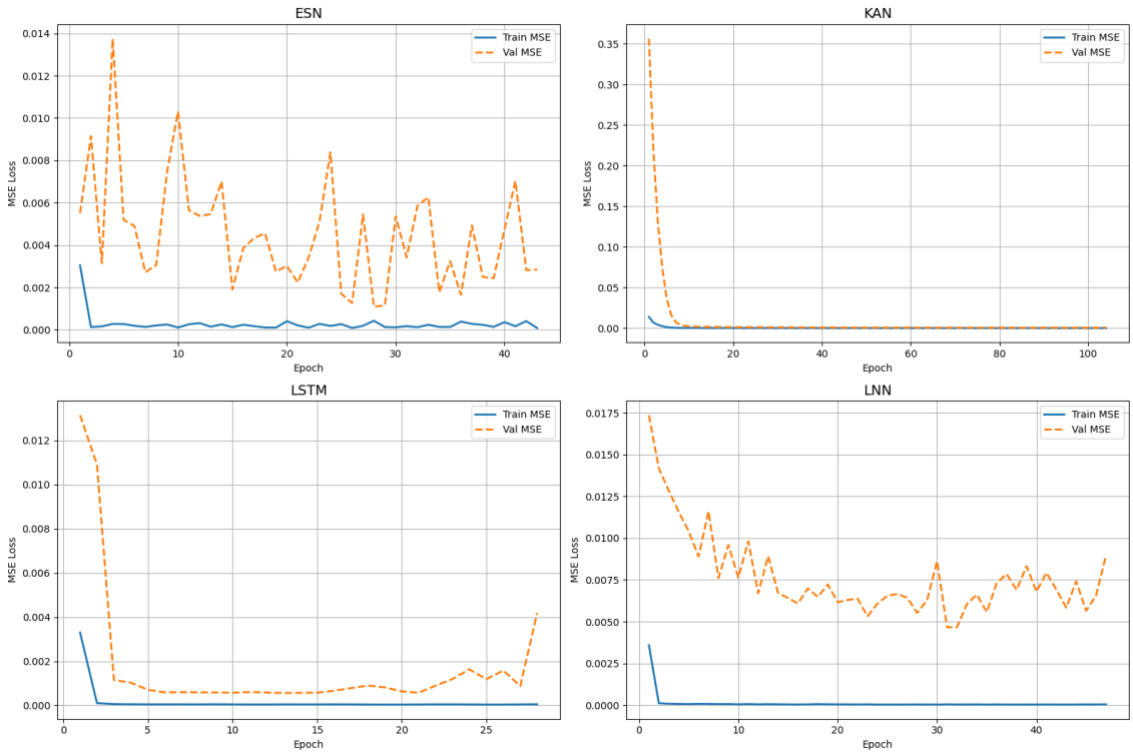

Figura 15. Curvas de pérdida de entrenamiento y validación de los modelos para Apple - Horizonte máximo.

Como se observa en la figura 15, KAN y LSTM muestran un descenso de pérdida muy limpio, lo que indica un entrenamiento estable y sin oscilaciones. El ESN alterna subidas y bajadas en su validación, reflejo de cierta irregularidad, y la LNN presenta variaciones constantes en el error de validación, señal de una convergencia más inestable.

4.1.3.4 Coste de entrenamiento y tamaño del modelo

Modelo	Épocas	Mejor época	Tiempo de entrenamiento (s)	Tiempo a mejor época (s)	Mejor Val MSE	Parámetros
ESN	43	28	42.49s	28.04s	0.001084	2001
KAN	104	89	44.09s	37.80s	0.000427	28
LSTM	28	13	12.36s	5.70s	0.000558	71297
LNN	47	32	151.07s	103.49s	0.004665	17921

Tabla 7. Coste de entrenamiento y tamaño del modelo para Apple - Horizonte máximo.

Según los datos de la tabla 7, la KAN sigue siendo el modelo más ligero (28 parámetros) y tarda alrededor de 44 s en llegar a su mejor validación. El ESN maneja 2 001 parámetros en 42 s, el LSTM entrena en solo 12 s pero con 71 297 parámetros, y la LNN, con 17 921 parámetros, necesita más de 2 minutos para estabilizarse en su mínima pérdida.

4.2 Microsoft (MSFT)

4.2.1 Horizonte de 5 años

4.2.1.1 Resumen de métricas

Modelo	MSE	RMSE	MAE	R^2	MAPE (%)	SMAPE (%)
ESN	870.99	29.51	28.26	-1.88	6.72%	6.98%
KAN	148.61	12.19	10.46	0.51	2.48%	2.52%
LSTM	347.29	18.64	16.37	-0.15	3.88%	3.97%
LNN	1140.63	33.77	32.69	-2.77	7.77%	8.10%

Tabla 8. Resumen de métricas de modelos para Microsoft - Horizonte de 5 años.

Según queda reflejado en la tabla 8, en este horizonte de cinco años para Microsoft, la KAN vuelve a liderar la precisión con un MAPE del 2,48 % y R^2=0,51, mientras que ESN y LNN registran errores elevados (6,72 % y 7,77 % respectivamente) y coeficientes negativos, indicativo de un pobre ajuste. El LSTM se sitúa en terreno intermedio, con un MAPE del 3,88 %.

4.2.1.2 Precio real vs predicción

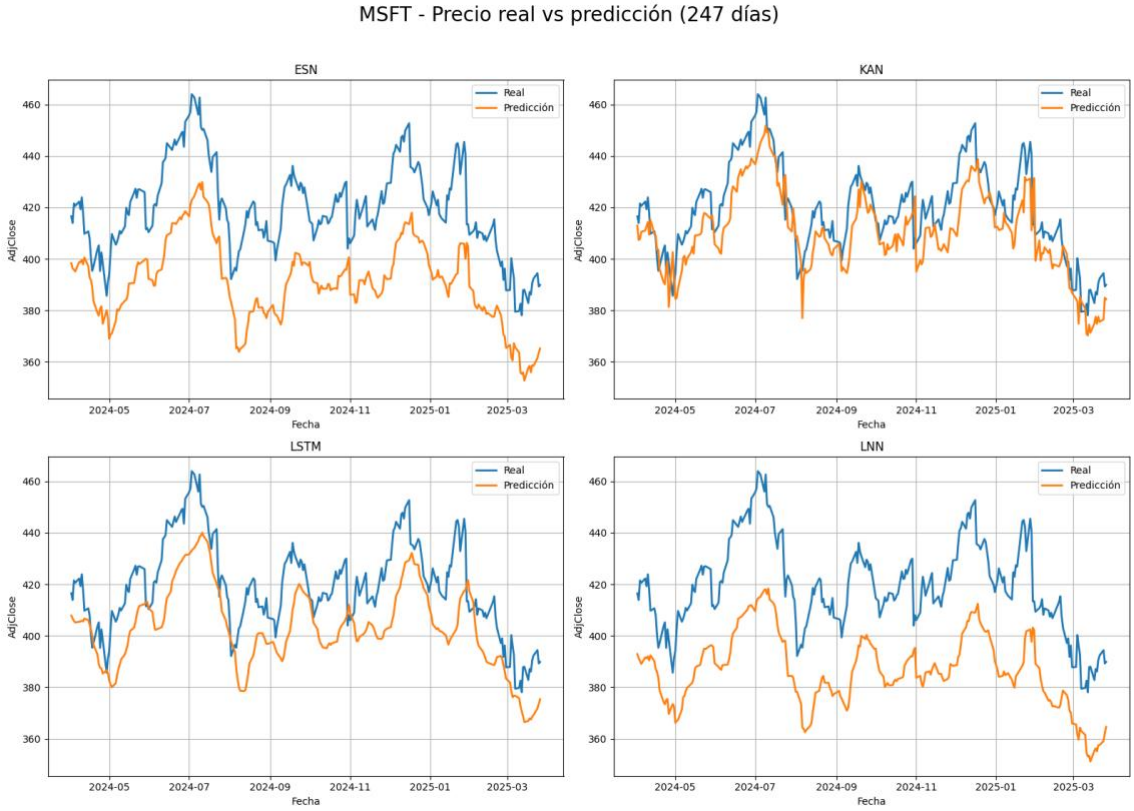

Figura 16. Precio real vs. predicción de los modelos para Microsoft - Horizonte de 5 años.

Como podemos observar en la figura 16, la KAN captura con relativa fidelidad las subidas y bajadas de la serie, aunque suaviza algo los extremos; el LSTM sigue la tendencia general, pero tiende a infravalorar los picos y valles, y ESN y LNN se quedan claramente por debajo en gran parte del recorrido.

4.2.1.3 Curvas de pérdida

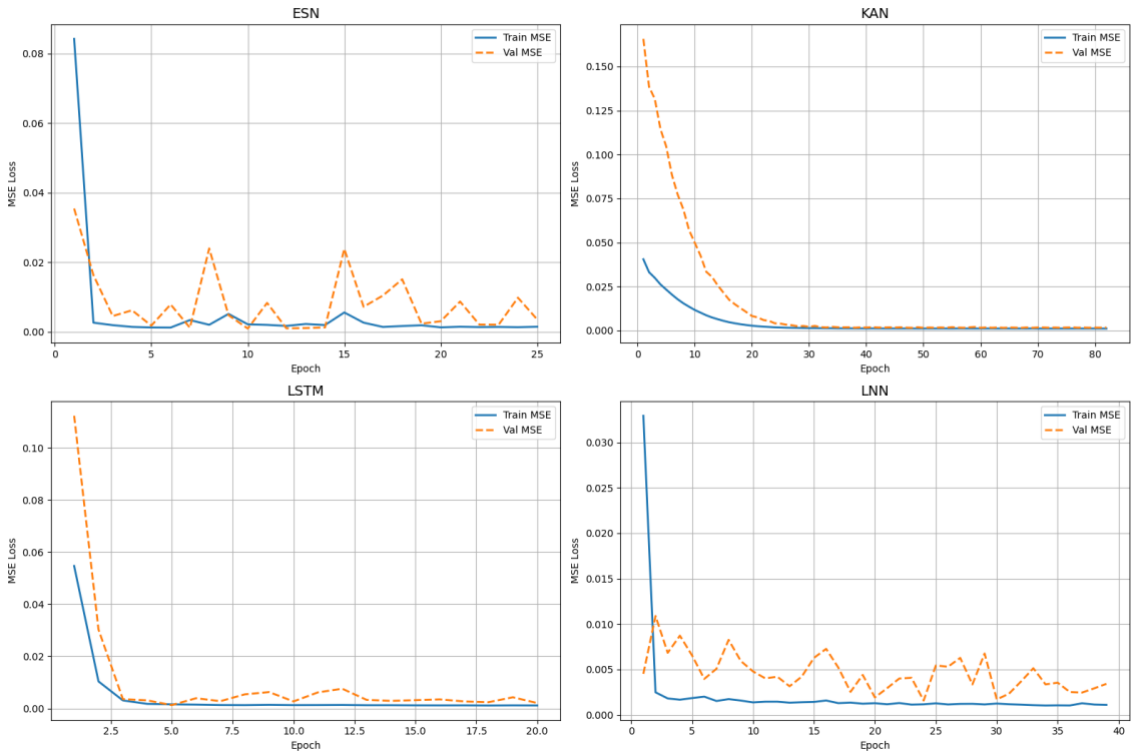

Figura 17. Curvas de pérdida de entrenamiento y validación de los modelos para Microsoft - Horizonte de 5 años.

Según los gráficos de la figura 17, KAN y LSTM muestran una disminución de la MSE de validación bastante estable, mientras que ESN y LNN exhiben fluctuaciones más marcadas en su pérdida de validación, señal de que les resulta más difícil ajustarse a la volatilidad del activo.

4.2.1.4 Coste de entrenamiento y tamaño del modelo

Modelo	Épocas	Mejor época	Tiempo de entrenamiento (s)	Tiempo a mejor época (s)	Mejor Val MSE	Parámetros
ESN	25	10	9.72s	3.96s	0.000974	2001
KAN	82	67	12.79s	10.60s	0.001498	28
LSTM	20	5	3.49s	1.15s	0.001271	71297
LNN	39	24	46.59s	28.88s	0.001576	17921

Tabla 9. Coste de entrenamiento y tamaño del modelo para Microsoft - Horizonte de 5 años.

Según los datos de la tabla 9, con solo 28 parámetros y entrenamiento en torno a 13 s, la KAN sigue destacando en ligereza; el LSTM, a pesar de sus casi 72 000 parámetros, converge en apenas 3 s, mientras que ESN y LNN requieren más tiempo y manejan mayor complejidad para un rendimiento inferior.

4.2.2 Horizonte de 10 años

4.2.2.1 Resumen de métricas

Modelo	MSE	RMSE	MAE	R^2	MAPE (%)	SMAPE (%)
ESN	636.26	25.22	21.45	0.83	5.44%	5.63%
KAN	85.24	9.23	7.61	0.98	2.09%	2.11%
LSTM	375.48	19.38	16.81	0.90	4.35%	4.47%
LNN	693.93	26.34	22.61	0.82	5.75%	5.97%

Tabla 10. Resumen de métricas de modelos para Microsoft - Horizonte de 10 años.

Como se muestra en la tabla 10, la KAN repite su dominio con un MAPE del 2,09 % y R^2=0,98, registrando el ajuste más fino. El LSTM le sigue en segundo lugar (MAPE 4,35 %, R^2=0,90), mientras que ESN y LNN ofrecen un desempeño más ajustado, con errores superiores al 5 %.

4.2.2.2 Precio real vs predicción

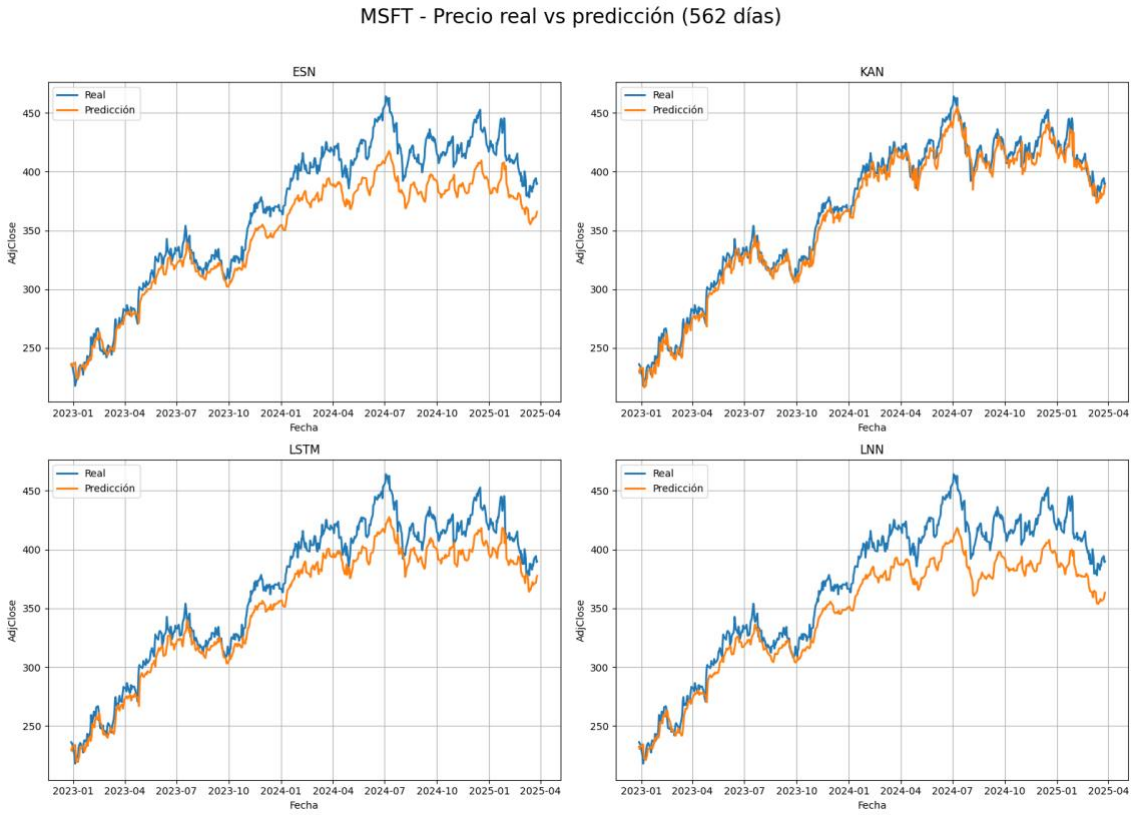

Figura 18. Precio real vs. predicción de los modelos para Microsoft - Horizonte de 10 años.

Como se observa en la gráfica de 562 días de la figura 18, la KAN se solapa prácticamente con la serie real, capturando las transiciones de tendencia. El LSTM mantiene la dirección general, pero atenúa los picos, el ESN tiende a infravalorar los máximos y la LNN produce una versión demasiado suavizada de la evolución

4.2.2.3 Curvas de pérdida

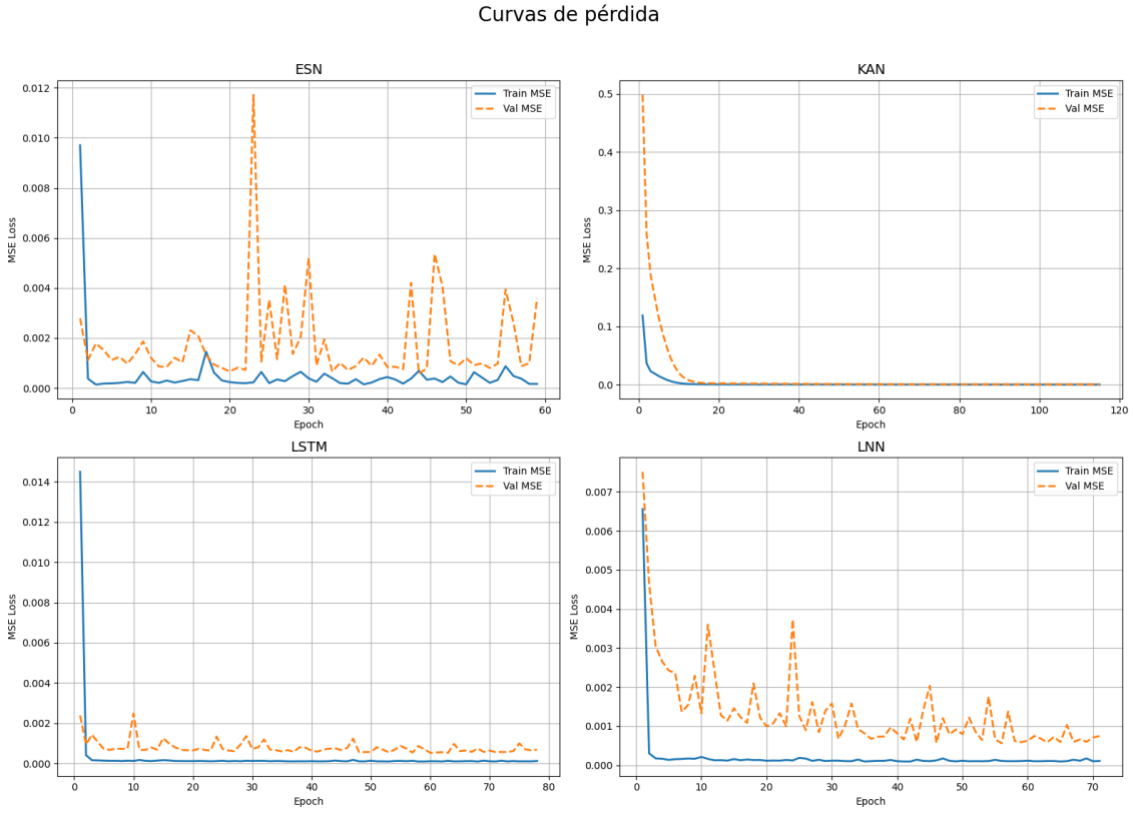

Figura 19. Curvas de pérdida de entrenamiento y validación de los modelos para Microsoft - Horizonte de 10 años.

Según las gráficas de la figura 19, KAN y LSTM muestran un descenso de la pérdida muy homogéneo y sin altibajos, reflejando un entrenamiento estable. En cambio, ESN y LNN experimentan oscilaciones en la pérdida de validación, indicando cierta irregularidad en la optimización.

4.2.2.4 Coste de entrenamiento y tamaño del modelo

Modelo	Épocas	Mejor época	Tiempo de entrenamiento (s)	Tiempo a mejor época (s)	Mejor Val MSE	Parámetros
ESN	59	44	46.21s	34.76s	0.000589	2001
KAN	115	100	38.47s	33.66s	0.000496	28
LSTM	78	63	25.33s	20.68s	0.000525	71297
LNN	71	56	163.03s	128.92s	0.000564	17921

Tabla 11. Coste de entrenamiento y tamaño del modelo para Microsoft - Horizonte de 10 años.

Según los datos de la tabla 11, la KAN sigue siendo la opción más ligera (28 parámetros) y necesita alrededor de 34 s para alcanzar su mínimo de validación. El LSTM, con alrededor de 71 000 parámetros, converge en unos 21 s, mientras que ESN y LNN requieren más tiempo de cómputo (35 s y 129 s) para ajustar sus respectivos 2 001 y 17 921 parámetros.

4.2.3 Horizonte máximo

4.2.3.1 Resumen de métricas

Modelo	MSE	RMSE	MAE	R^2	MAPE (%)	SMAPE (%)
ESN	1041.96	32.28	26.46	0.81	7.04%	7.37%
KAN	77.22	8.79	7.14	0.99	2.16%	2.18%
LSTM	451.89	21.26	17.44	0.92	4.72%	4.86%
LNN	272.59	16.51	13.62	0.95	3.76%	3.84%

Tabla 12. Resumen de métricas de modelos para Microsoft - Horizonte máximo.

Según los datos de la tabla 12, con el histórico completo, la KAN confirma su superioridad con un MAPE del 2,16 % y un R^2 de 0,99, capturando casi toda la variabilidad real. Aunque la LNN obtiene el RMSE más bajo (16,51), su MAPE (3,76 %) sigue por detrás de KAN, mientras que LSTM y ESN presentan errores mayores (4,72 % y 7,04 %, respectivamente).

4.2.3.2 Precio real vs predicción

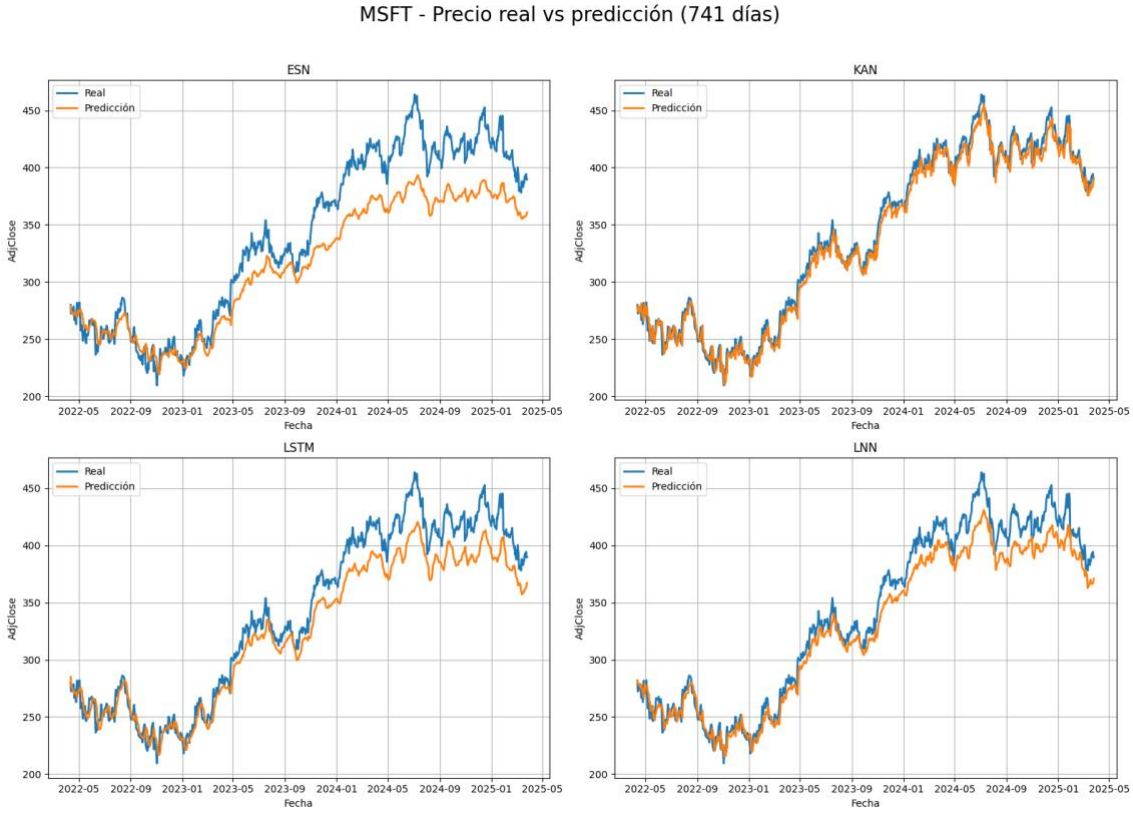

Figura 20. Precio real vs. predicción de los modelos para Microsoft - Horizonte máximo.

Tal y como queda reflejado en los 741 días de la figura 20, la curva de KAN se solapa prácticamente con la serie real, incluso en los picos de mediados de 2024. El LSTM reproduce bien la tendencia general, pero atenúa ligeramente las subidas, el ESN tiende a subestimar los máximos, y la LNN ofrece un ajuste más suave y conservador.

4.2.3.3 Curvas de pérdida

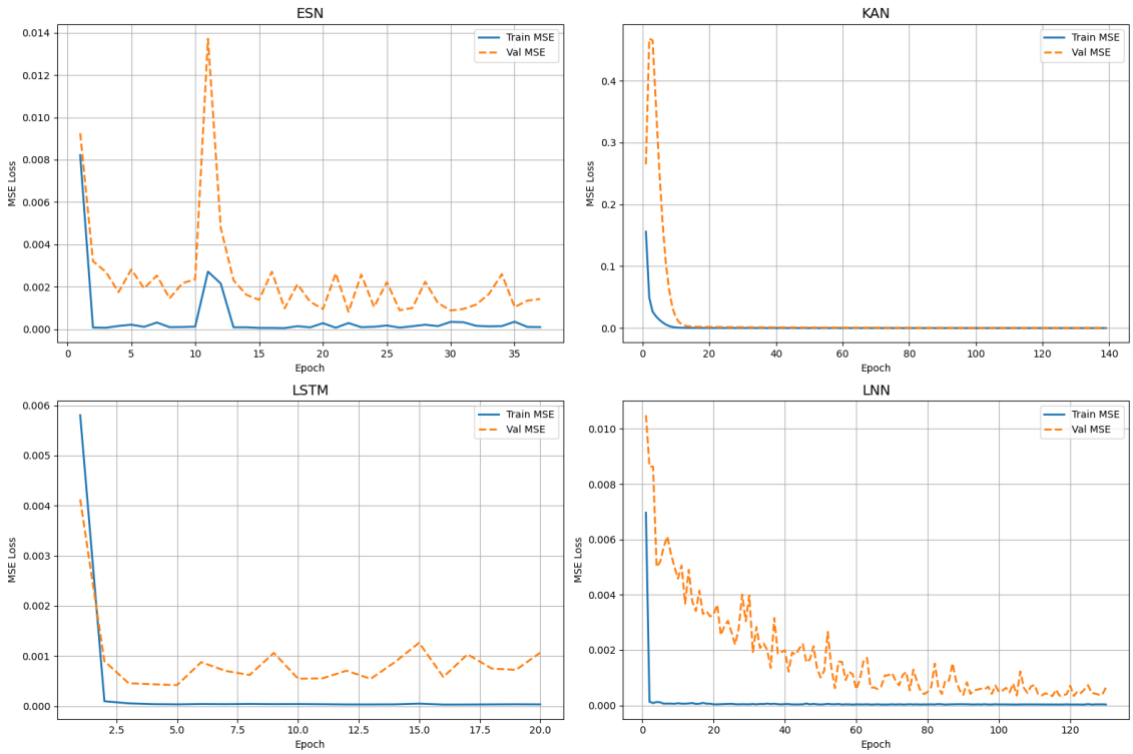

Figura 21. Curvas de pérdida de entrenamiento y validación de los modelos para Microsoft - Horizonte máximo.

Como se observa en la figura 21, KAN y LSTM exhiben un descenso muy ordenado y continuo en su MSE de validación, reflejo de un entrenamiento estable. Por contra, ESN y LNN muestran varios repuntes en la pérdida de validación, señal de que les cuesta mantener la consistencia a lo largo de todo el proceso.

4.2.3.4 Coste de entrenamiento y tamaño del modelo

Modelo	Épocas	Mejor época	Tiempo de entrenamiento (s)	Tiempo a mejor época (s)	Mejor Val MSE	Parámetros
ESN	37	22	39.74s	24.73s	0.000823	2001
KAN	139	124	58.74s	52.84s	0.000282	28
LSTM	20	5	10.68s	2.71s	0.000418	71297
LNN	130	115	397.07s	352.80s	0.000333	17921

Tabla 13. Coste de entrenamiento y tamaño del modelo para Microsoft - Horizonte máximo.

Según los datos de la tabla 13, la KAN vuelve a destacar por su ligereza (28 parámetros) y tarda unos 53 s en llegar a su mejor validación. El ESN maneja 2 001 parámetros en 40 s, el LSTM, con 71 297 parámetros, converge muy rápido (≈10 s), y la LNN, con 17 921 parámetros, necesita más de 6 min para estabilizarse en su menor pérdida.

4.3 Amazon (AMZN)

4.3.1 Horizonte de 5 años

4.3.1.1 Resumen de métricas

Modelo	MSE	RMSE	MAE	R^2	MAPE (%)	SMAPE (%)
ESN	64.39	8.02	6.41	0.82	3.14%	3.20%
KAN	86.88	9.32	7.56	0.76	3.71%	3.80%
LSTM	45.85	6.77	5.36	0.88	2.65%	2.69%
LNN	92.65	9.63	7.54	0.75	3.65%	3.74%

Tabla 14. Resumen de métricas de modelos para Amazon - Horizonte de 5 años.

Tal y como refleja la tabla 14, en el periodo de cinco años la LSTM lidera con un MAPE del 2,65 % y R^2=0,88, capturando muy bien la dinámica de Amazon. El ESN también rinde bien (MAPE 3,14 %, R^2=0,82), mientras que KAN y LNN quedan un poco por detrás, con errores del 3,71 % y 3,65 % respectivamente.

4.3.1.2 Precio real vs predicción

Figura 22. Precio real vs. predicción de los modelos para Amazon - Horizonte de 5 años.

Como se observa en la figura 22, la LSTM se superpone casi por completo a la serie real y reproduce fielmente los picos de enero-febrero. El ESN mantiene la dirección general con ligera subestimación en los máximos, y tanto KAN como LNN suavizan un poco las oscilaciones más fuertes.

4.3.1.3 Curvas de pérdida

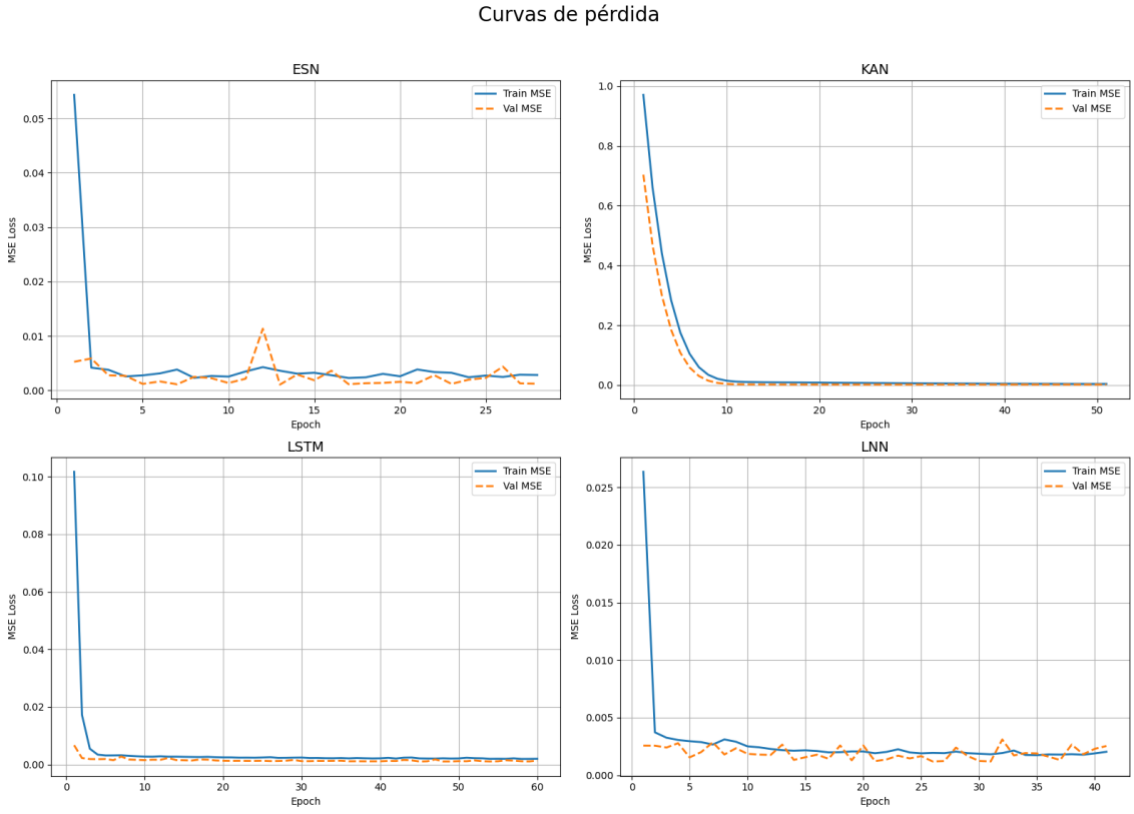

Figura 23. Curvas de pérdida de entrenamiento y validación de los modelos para Amazon - Horizonte de 5 años.

Según los gráficos de la figura 23, KAN y LSTM muestran una caída de la MSE muy progresiva y sin altibajos, reflejo de un entrenamiento muy estable. En cambio, ESN experimenta algunos repuntes esporádicos en validación y la LNN registra pequeñas variaciones a lo largo del proceso.

4.3.1.4 Coste de entrenamiento y tamaño del modelo

Modelo	Épocas	Mejor época	Tiempo de entrenamiento (s)	Tiempo a mejor época (s)	Mejor Val MSE	Parámetros
ESN	28	13	10.54s	5.02s	0.00107	2001
KAN	51	36	7.84s	5.50s	0.001838	28
LSTM	60	45	12.49s	9.41s	0.001021	71297
LNN	41	26	47.53s	30.20s	0.001186	17921

Tabla 15. Coste de entrenamiento y tamaño del modelo para Amazon - Horizonte de 5 años.

Como reflejan los datos de la tabla 15, la KAN destaca por su ligereza (28 parámetros) y rapidez (≈8 s). El ESN, con 2 001 parámetros, también es ágil (≈5 s), mientras que la LSTM, pese a sus casi 72 000 parámetros, converge en torno a 12 s. La LNN, con 17 921 parámetros, es la más costosa de entrenar (≈47 s).

4.3.2 Horizonte de 10 años

4.3.2.1 Resumen de métricas

Modelo	MSE	RMSE	MAE	R^2	MAPE (%)	SMAPE (%)
ESN	50.63	7.12	5.35	0.97	3.12%	3.18%
KAN	17.55	4.19	3.14	0.99	2.02%	2.04%
LSTM	27.07	5.20	3.82	0.98	2.32%	2.34%
LNN	30.72	5.54	4.13	0.98	2.49%	2.53%

Tabla 16. Resumen de métricas de modelos para Amazon - Horizonte de 10 años.

Según los datos del horizonte de diez años en la tabla 16, la KAN repite su dominio con un MAPE del 2,02 % y R^2=0,99, capturando casi con exactitud la serie histórica. El LSTM le sigue muy de cerca (MAPE 2,32 %, R^2=0,98), mientras que ESN y LNN también rinden bien (MAPE 3,12 % y 2,49 %, ambos R^2≈0,97-0,98).

4.3.2.2 Precio real vs predicción

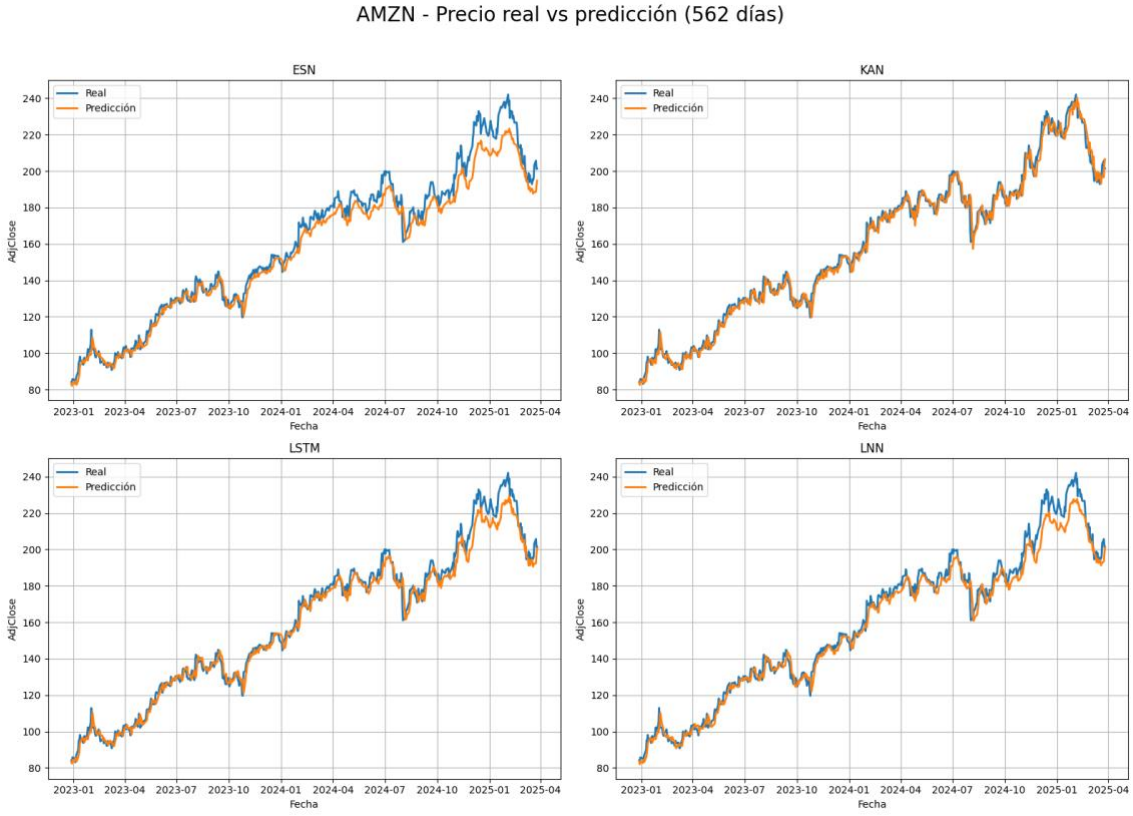

Figura 24. Precio real vs. predicción de los modelos para Amazon - Horizonte de 10 años.

Como se observa en la figura 24, la predicción de KAN se solapa casi perfectamente con la serie real a lo largo de los 562 días. LSTM muestra un ligero suavizado de los picos, ESN tiende a infravalorar algunos máximos y LNN, aunque bastante cercana, mantiene un ajuste algo más conservador en los picos.

4.3.2.3 Curvas de pérdida

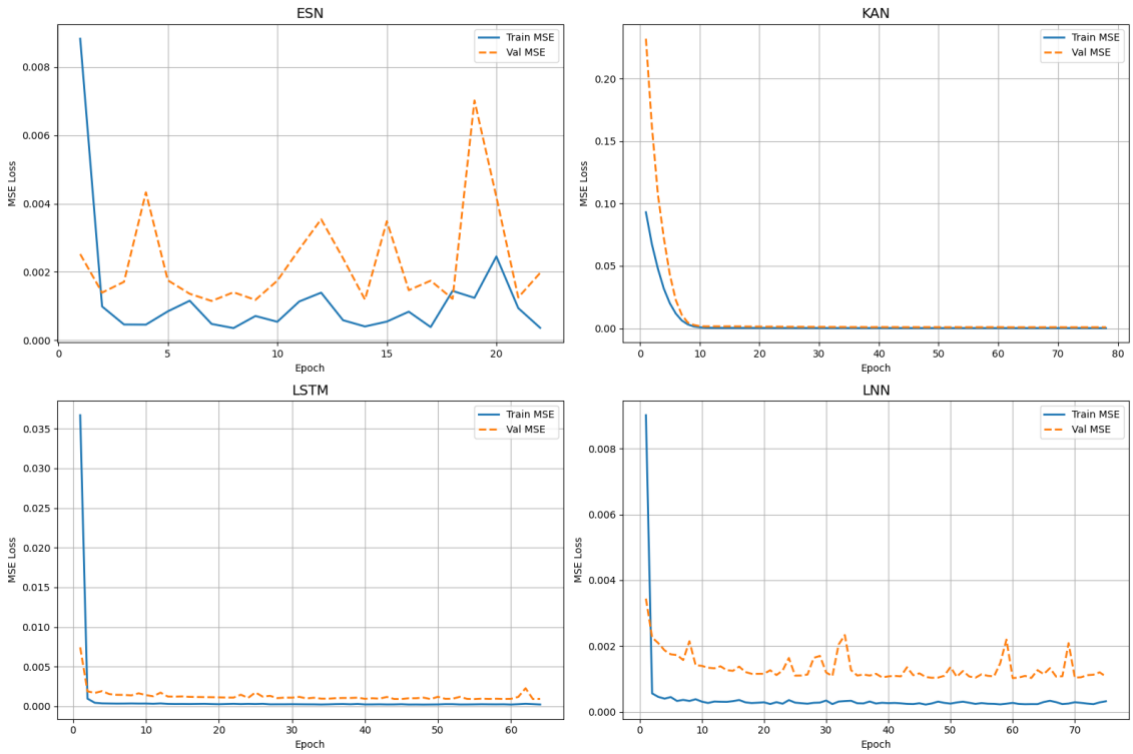

Figura 25. Curvas de pérdida de entrenamiento y validación de los modelos para Amazon - Horizonte de 10 años.

Tal y como se refleja en la figura 25, KAN y LSTM presentan un descenso ordenado y mantenido de la MSE en entrenamiento y validación, lo que refleja un ajuste muy estable. ESN y LNN exhiben pequeños repuntes en la pérdida de validación, señal de mayor sensibilidad a ciertos ciclos de entrenamiento.

4.3.2.4 Coste de entrenamiento y tamaño del modelo

Modelo	Épocas	Mejor época	Tiempo de entrenamiento (s)	Tiempo a mejor época (s)	Mejor Val MSE	Parámetros
ESN	22	7	16.10s	5.20s	0.001153	2001
KAN	78	63	24.49s	19.55s	0.000893	28
LSTM	64	49	26.82s	20.55s	0.00092	71297
LNN	75	60	181.68s	145.63s	0.001024	17921

Tabla 17. Coste de entrenamiento y tamaño del modelo para Amazon - Horizonte de 10 años.

Según los datos de la tabla 17, la KAN, con sólo 28 parámetros, converge en unos 20 s, lo que la sitúa como la opción más ligera. El ESN maneja 2 001 parámetros y requiere alrededor de 5 s, el LSTM ajusta 71 297 parámetros en unos 21 s, y la LNN, con 17 921 parámetros, necesita cerca de 146 s para alcanzar su mínima pérdida.

4.3.3 Horizonte máximo

4.3.3.1 Resumen de métricas

Modelo	MSE	RMSE	MAE	R^2	MAPE (%)	SMAPE (%)
ESN	210.62	14.51	10.62	0.87	6.40%	6.60%
KAN	20.05	4.48	3.36	0.99	2.40%	2.39%
LSTM	43.95	6.63	5.22	0.97	3.72%	3.67%
LNN	63.31	7.96	5.82	0.96	3.68%	3.71%

Tabla 18. Resumen de métricas de modelos para Amazon - Horizonte máximo.

Como se muestra en la tabla 18, la KAN vuelve a imponerse con un MAPE de 2,40 % y R^2=0,99, mostrando un ajuste casi impecable en el histórico completo. El LSTM le sigue en precisión (MAPE 3,72 %, R^2=0,97), mientras que LNN (3,68 %, R^2=0,96) y ESN (6,40 %, R^2=0,87) presentan un rendimiento algo inferior.

4.3.3.2 Precio real vs predicción

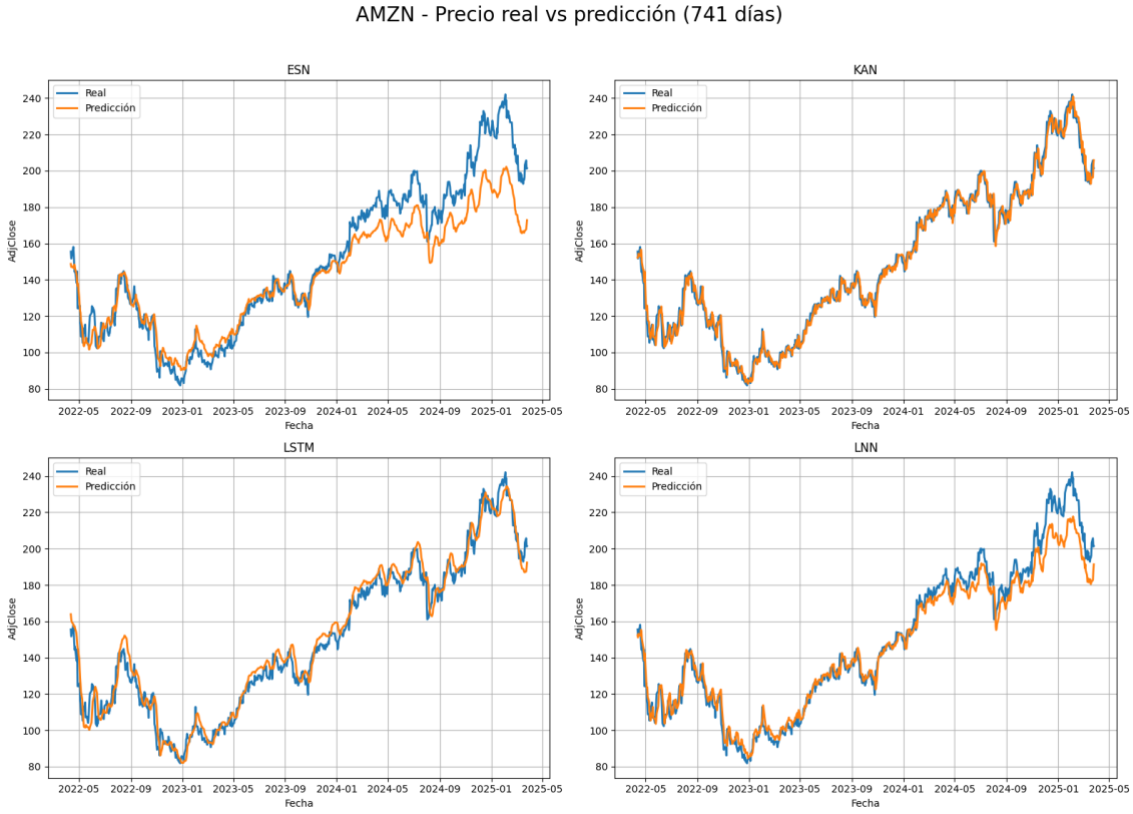

Figura 26. Precio real vs. predicción de los modelos para Amazon - Horizonte máximo.

Como se refleja en las gráficas de la figura 26, la KAN reproduce casi al milímetro los movimientos de la serie real a lo largo de los 741 días. El LSTM captura muy bien las tendencias, pero atenúa ligeramente los picos, la LNN ofrece un ajuste relativamente fiel con algo más de suavizado, y el ESN tiende a subestimar los máximos alcistas.

4.3.3.3 Curvas de pérdida

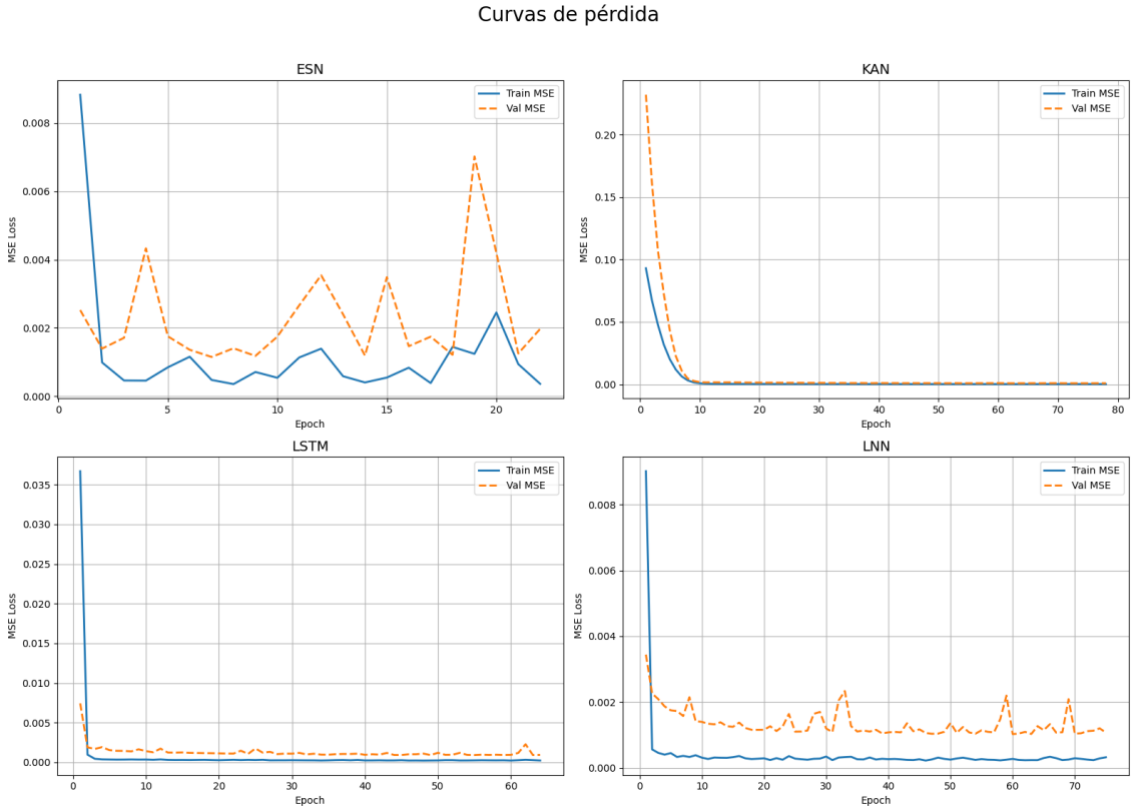

Figura 27. Curvas de pérdida de entrenamiento y validación de los modelos para Amazon - Horizonte máximo.

Tal y como queda reflejado en la figura 27, KAN y LSTM muestran una caída de la pérdida de validación muy limpia y sostenida, reflejo de un entrenamiento estable. ESN presenta picos esporádicos en su MSE de validación y la LNN mantiene oscilaciones prolongadas, señal de una convergencia menos consistente.

4.3.3.4 Coste de entrenamiento y tamaño del modelo

Modelo	Épocas	Mejor época	Tiempo de entrenamiento (s)	Tiempo a mejor época (s)	Mejor Val MSE	Parámetros
ESN	22	7	22.76s	7.33s	0.002395	2001
KAN	110	95	43.74s	37.19s	0.00054	28
LSTM	17	2	8.93s	1.00s	0.001465	71297
LNN	103	88	316.69s	270.91s	0.000676	17921

Tabla 19. Coste de entrenamiento y tamaño del modelo para Amazon - Horizonte máximo.

Según los datos de la tabla 19, la KAN, con solo 28 parámetros, entrena en unos 44 s y mantiene la mejor validación tras 95 épocas. El ESN maneja 2 001 parámetros y completa su ajuste en unos 23 s; el LSTM, a pesar de sus 71 297 parámetros, converge rápidamente en menos de 9 s; y la LNN, con 17 921 parámetros, necesita más de 5 min para estabilizarse.

4.4 Nvidia (NVDA)

4.4.1 Horizonte de 5 años

4.4.1.1 Resumen de métricas

Modelo	MSE	RMSE	MAE	R^2	MAPE (%)	SMAPE (%)
ESN	1869.30	43.24	41.38	-4.96	33.30%	40.30%
KAN	38.03	6.17	4.82	0.88	3.99%	4.02%
LSTM	700.64	26.47	24.73	-1.23	19.72%	22.09%
LNN	2754.20	52.48	50.71	-7.78	41.08%	52.11%

Tabla 20. Resumen de métricas de modelos para Nvidia - Horizonte de 5 años.

Tal y como aparece este horizonte de cinco años en la tabla 20, la KAN vuelve a destacar con un MAPE de 3,99 % y un R^2 de 0,88, consiguiendo el mejor ajuste sin caer en el sobreajuste. El LSTM queda bastante por detrás (MAPE 19,72 %, R^2 negativo) y tanto ESN como LNN pierden completamente la capacidad predictiva (R^2 de -4,96 y -7,78, respectivamente), reflejo de la enorme volatilidad de Nvidia.

4.4.1.2 Precio real vs predicción

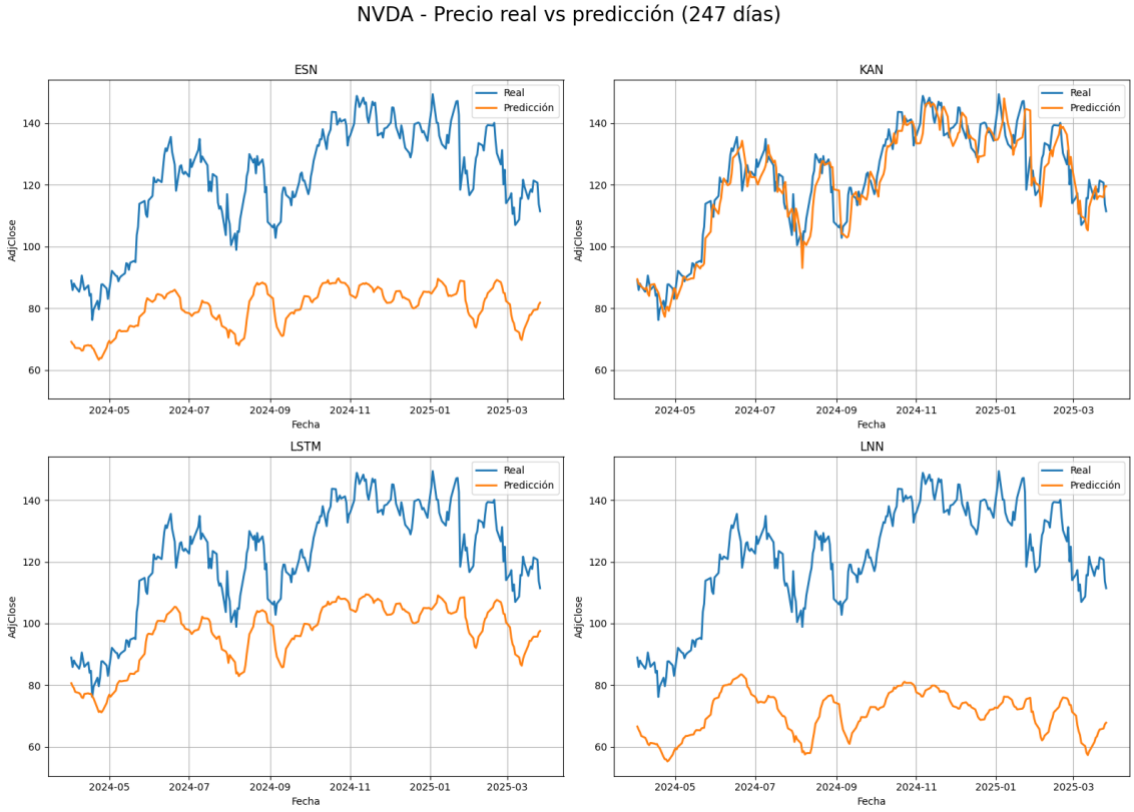

Figura 28. Precio real vs. predicción de los modelos para Nvidia - Horizonte de 5 años.

Como se observa en la figura 28, la curva de KAN sigue la serie histórica con bastante fidelidad, reproduciendo los altibajos, aunque suaviza un poco los picos. El LSTM capta la dirección general, pero subestima sistemáticamente los máximos, mientras que ESN y LNN ofrecen una predicción demasiado conservadora y plana.

4.4.1.3 Curvas de pérdida

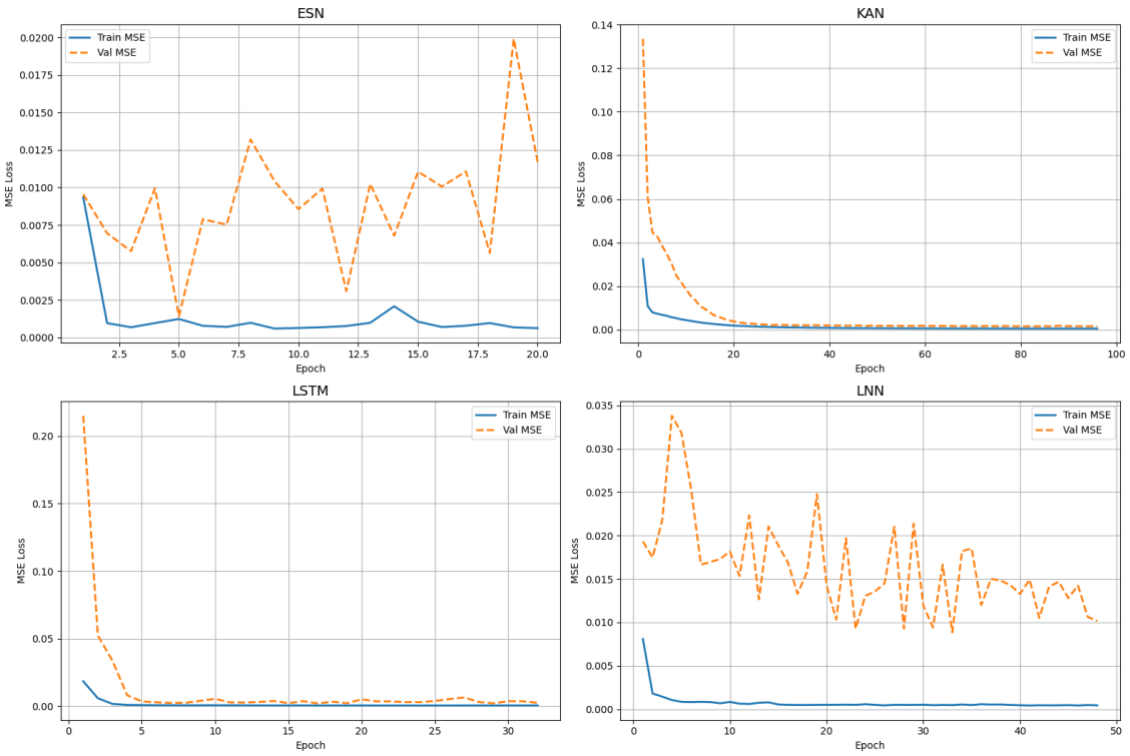

Figura 29. Curvas de pérdida de entrenamiento y validación de los modelos para Nvidia - Horizonte de 5 años.

Según la figura 29, KAN y LSTM presentan una caída ordenada de la MSE, con validación muy estable. En cambio, ESN muestra varios repuntes en la pérdida de validación y LNN mantiene un nivel alto y errático de MSE, señal de que le cuesta adaptarse a los saltos bruscos del precio.

4.4.1.4 Coste de entrenamiento y tamaño del modelo

Modelo	Épocas	Mejor época	Tiempo de entrenamiento (s)	Tiempo a mejor época (s)	Mejor Val MSE	Parámetros
ESN	20	5	7.52s	2.09s	0.001412	2001
KAN	96	81	15.88s	13.35s	0.001483	28
LSTM	32	17	6.51s	3.48s	0.002035	71297
LNN	48	33	57.16s	39.25s	0.008846	17921

Tabla 21. Coste de entrenamiento y tamaño del modelo para Nvidia - Horizonte de 5 años.

Según los datos de la tabla 21, La KAN, con solo 28 parámetros y unos 16 s de entrenamiento, ofrece el mejor compromiso entre ligereza y precisión. El ESN, con 2 001 parámetros, tarda menos de 8 s pero rinde peor; el LSTM, a pesar de sus 71 297 parámetros, converge en 6 s; y la LNN, con 17 921 parámetros, requiere casi un minuto para ajustar su alta complejidad.

4.4.2 Horizonte de 10 años

4.4.2.1 Resumen de métricas

Modelo	MSE	RMSE	MAE	R^2	MAPE (%)	SMAPE (%)
ESN	1828.20	42.76	31.72	-0.01	30.52%	39.05%
KAN	18.23	4.27	2.87	0.99	3.59%	3.62%
LSTM	784.40	28.01	20.03	0.57	18.63%	21.76%
LNN	1290.61	35.93	25.31	0.29	22.92%	28.25%

Tabla 22. Resumen de métricas de modelos para Nvidia - Horizonte de 10 años.

Como queda reflejado en los datos de la tabla 22, la KAN destaca con un MAPE del 3,59 % y R^2=0,99, capturando casi toda la variabilidad histórica. El LSTM y la LNN quedan muy por detrás (errores del 18-23 % y R^2 bajos), mientras que el ESN prácticamente no consigue ajustarse (MAPE 30,5 %, $R^2 \approx 0$).

4.4.2.2 Precio real vs predicción

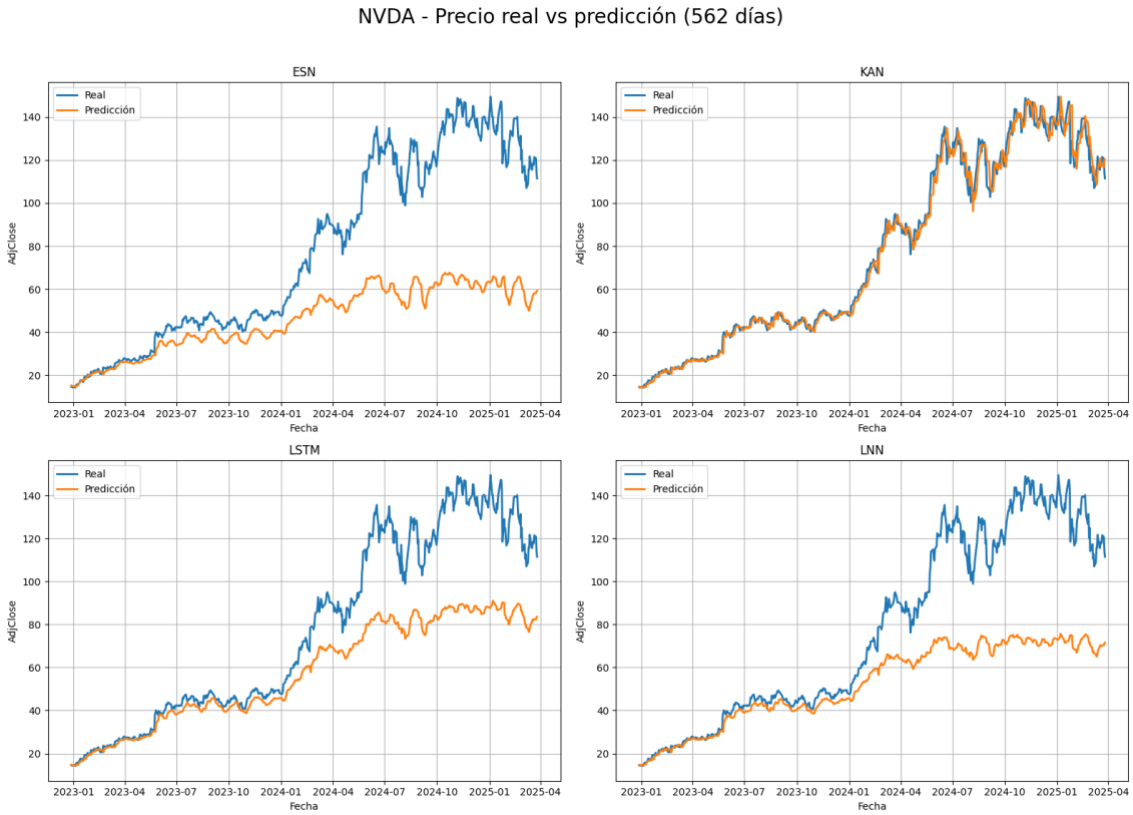

Figura 30. Precio real vs. predicción de los modelos para Nvidia - Horizonte de 10 años.

Como vemos en la figura 30 del gráfico de diez años, la KAN se superpone casi por completo a la serie real, incluso en los movimientos abruptos de 2024. LSTM y LNN subestiman de manera acusada las zonas alcistas y ESN apenas consigue seguir la tendencia.

4.4.2.3 Curvas de pérdida

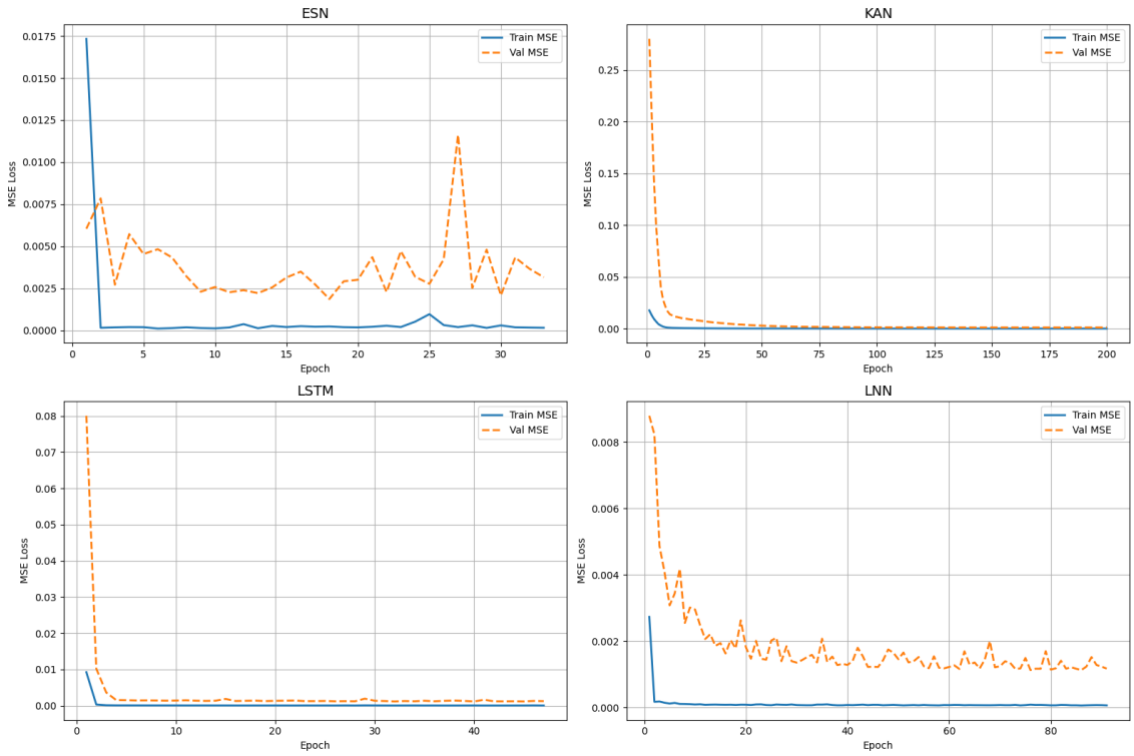

Figura 31. Curvas de pérdida de entrenamiento y validación de los modelos para Nvidia - Horizonte de 10 años.

Tal y como refleja la figura 31, KAN muestra un descenso limpio y sostenido de la MSE de validación, reflejo de un entrenamiento sólido. ESN y LNN presentan variaciones continuas en su validación, y LSTM cae rápido al principio, pero luego mantiene un nivel de error estable más alto.

4.4.2.4 Coste de entrenamiento y tamaño del modelo

Modelo	Épocas	Mejor época	Tiempo de entrenamiento (s)	Tiempo a mejor época (s)	Mejor Val MSE	Parámetros
ESN	33	18	27.22s	15.27s	0.00186	2001
KAN	200	200	61.62s	61.62s	0.001102	28
LSTM	47	32	19.49s	13.46s	0.001178	71297
LNN	91	76	225.70s	186.20s	0.001124	17921

Tabla 23. Resumen de métricas de modelos para Nvidia - Horizonte de 10 años.

Según los datos mostrados en la tabla 23, con solo 28 parámetros, la KAN tarda alrededor de un minuto en afinarse, ofreciendo el mejor balance ligereza-precisión. El ESN es rápido pero ineficaz, el LSTM necesita menos de 20 s, pero maneja decenas de miles de parámetros, y la LNN requiere varios minutos para llegar a su mínima pérdida.

4.4.3 Horizonte máximo

4.4.3.1 Resumen de métricas

Modelo	MSE	RMSE	MAE	R^2	MAPE (%)	SMAPE (%)
ESN	1404.56	37.48	24.89	0.32	25.69%	32.46%
KAN	13.57	3.68	2.31	0.99	3.77%	3.78%
LSTM	922.50	30.37	19.02	0.56	18.62%	22.45%
LNN	2898.10	53.83	37.81	-0.40	42.66%	61.07%

Tabla 24. Resumen de métricas de modelos para Nvidia - Horizonte máximo.

Como se muestra en la tabla 24, la KAN mantiene el mejor ajuste con un MAPE de 3,77 % y R^2=0,99, mientras que ESN y LNN pierden precisión (MAPE 25,69 % y 42,66 %) y LSTM queda en un término medio (MAPE 18,62 % y R^2=0,56).

4.4.3.2 Precio real vs predicción

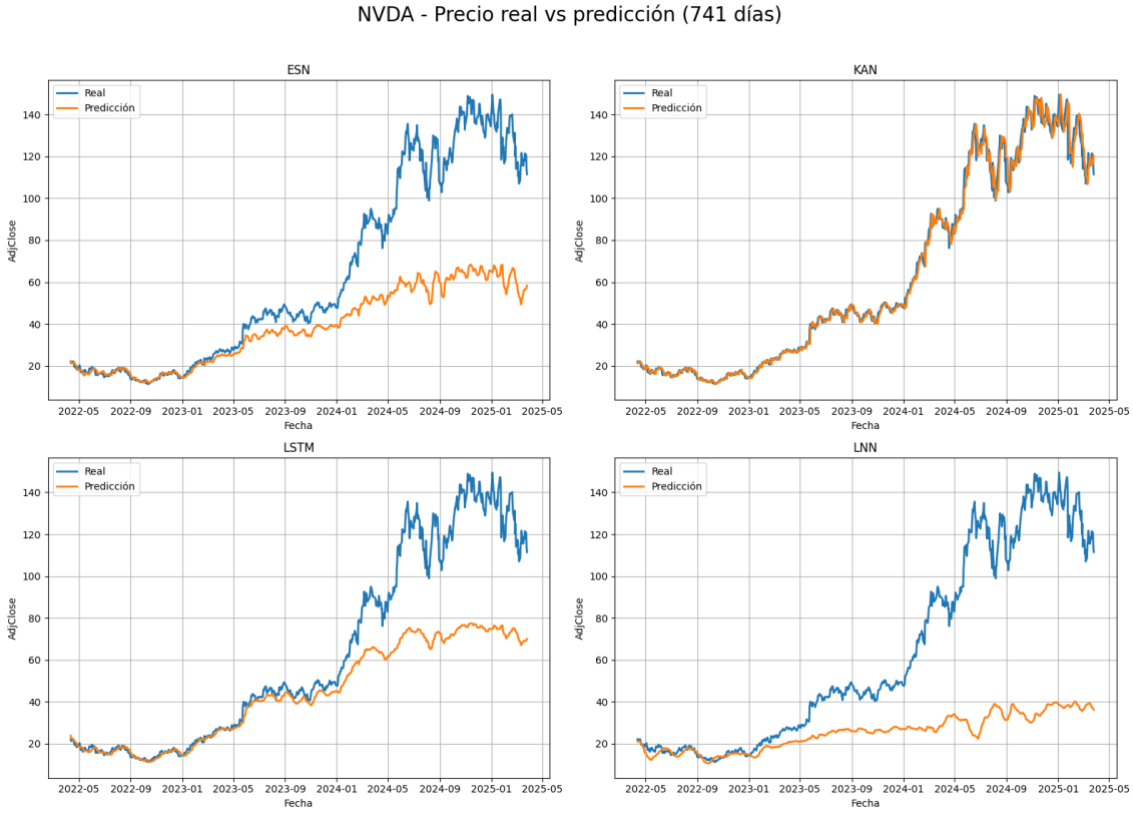

Figura 32. Precio real vs. predicción de los modelos para Nvidia - Horizonte máximo.

Como se observa en la figura 32, KAN reproduce prácticamente todos los movimientos del precio real, LSTM atenúa los picos, ESN subestima de forma constante y LNN ofrece un recorrido demasiado plano.

4.4.3.3 Curvas de pérdida

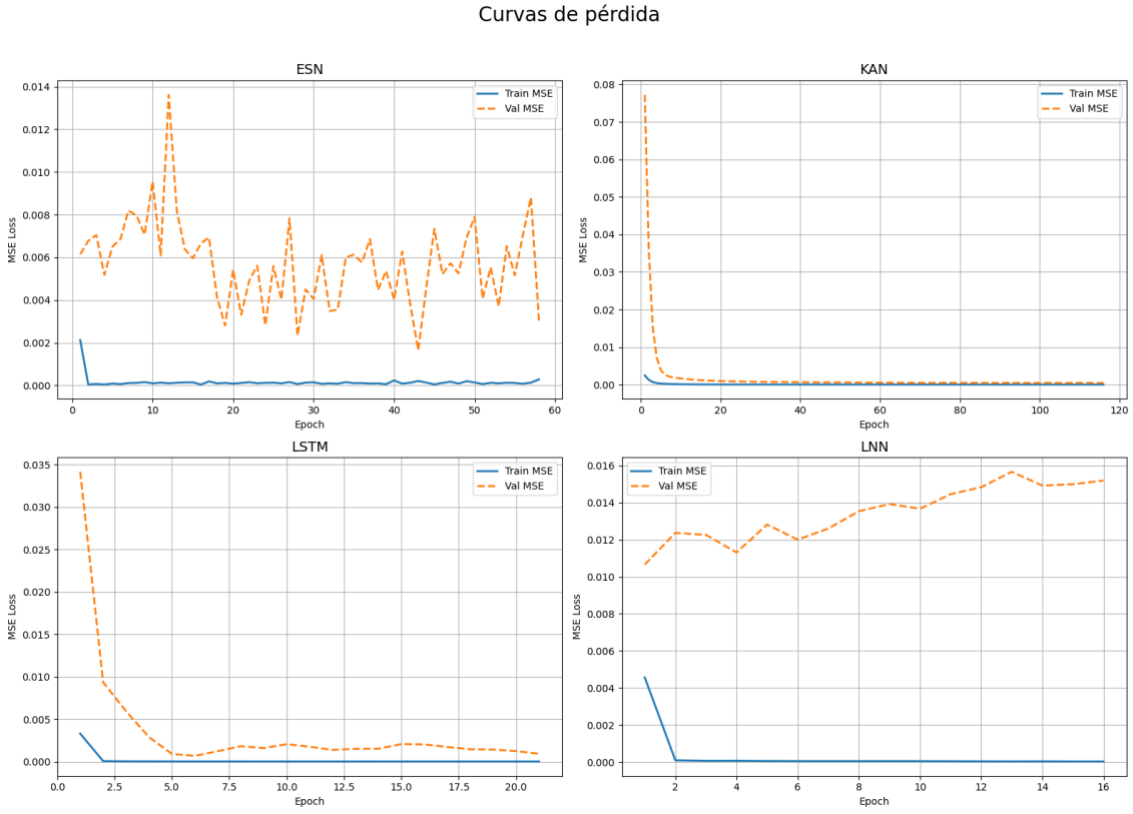

Figura 33. Curvas de pérdida de entrenamiento y validación de los modelos para Nvidia - Horizonte máximo.

Según el gráfico de la figura 33, KAN y LSTM muestran una disminución de la pérdida de validación muy limpia y sostenida; ESN y LNN alternan varios repuntes en validación, reflejo de mayor inestabilidad.

4.4.3.4 Coste de entrenamiento y tamaño del modelo

Modelo	Épocas	Mejor época	Tiempo de entrenamiento (s)	Tiempo a mejor época (s)	Mejor Val MSE	Parámetros
ESN	58	43	58.11s	43.37s	0.001649	2001
KAN	116	101	46.88s	40.86s	0.000463	28
LSTM	21	6	11.33s	3.27s	0.000682	71297
LNN	16	1	49.39s	3.17s	0.010662	17921

Tabla 25. Coste de entrenamiento y tamaño del modelo para Nvidia - Horizonte máximo.

Según los datos de la tabla 25, con solo 28 parámetros y menos de un minuto de entrenamiento, la KAN es la opción más eficiente. ESN maneja 2 001 parámetros en alrededor de 58 s, LSTM ajusta decenas de miles de parámetros en 11 s, y LNN necesita cerca de 49 s para llegar a su menor pérdida.

4.5 Meta (META)

4.5.1 Horizonte de 5 años

4.5.1.1 Resumen de métricas

Modelo	MSE	RMSE	MAE	R^2	MAPE (%)	SMAPE (%)
ESN	1035.97	32.19	26.89	0.79	4.67%	4.81%
KAN	252.15	15.88	12.50	0.95	2.27%	2.28%
LSTM	1715.40	41.42	35.88	0.65	6.18%	6.42%
LNN	2729.53	52.24	44.93	0.44	7.70%	8.08%

Tabla 26. Resumen de métricas de modelos para Meta - Horizonte de 5 años.

Como vemos reflejado en la taba 26 de este horizonte de cinco años, la KAN vuelve a imponerse con un MAPE del 2,27 % y un R^2 de 0,95, ofreciendo un ajuste muy preciso. El ESN rinde de forma aceptable (MAPE 4,67 %, R^2 = 0,79), mientras que LSTM y LNN presentan un desempeño inferior, con errores por encima del 6 % y coeficientes de determinación más bajos (R^2 = 0,65 y 0,44, respectivamente).

4.5.1.2 Precio real vs predicción

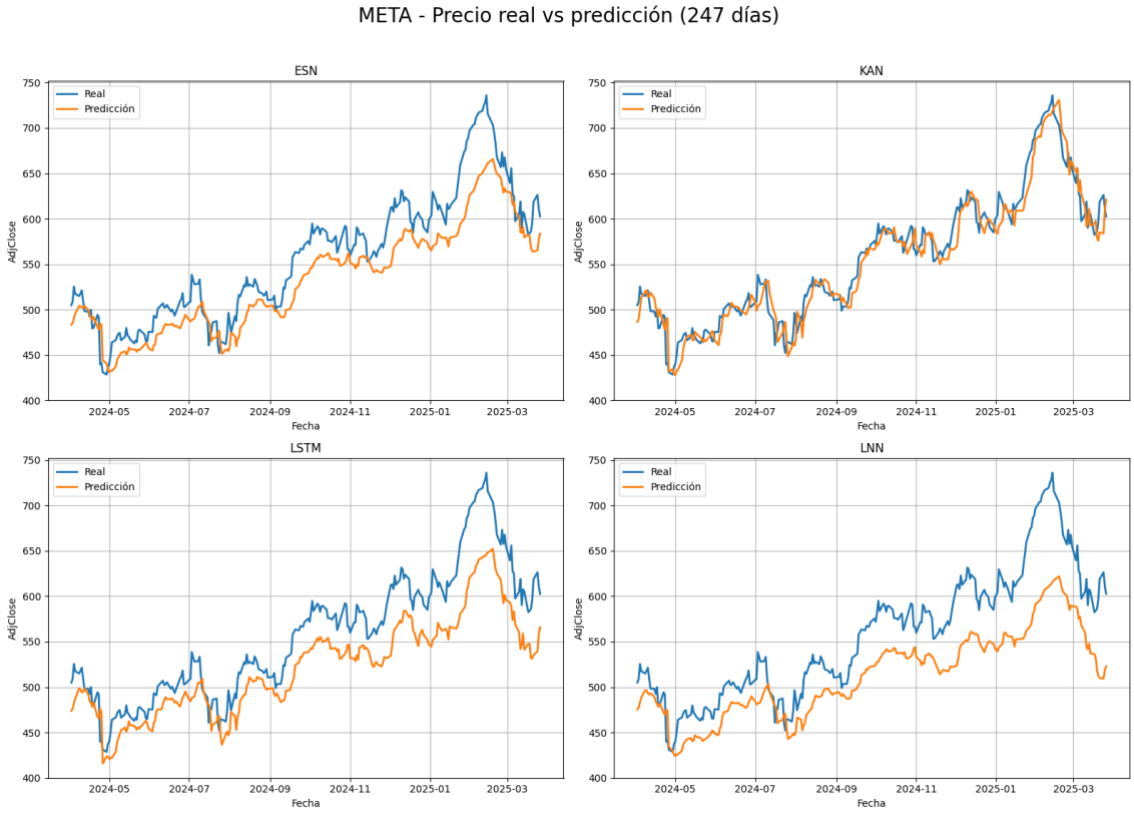

Figura 34. Precio real vs. predicción de los modelos para Meta - Horizonte de 5 años.

Como se observa en la figura 34, la KAN casi replica punto a punto la serie histórica, incluso en los retrocesos de inicios de 2025. El ESN capta la tendencia general, pero tiende a subestimar los picos más marcados; LSTM suaviza aún más las oscilaciones y la LNN ofrece una versión muy conservadora del movimiento.

4.5.1.3 Curvas de pérdida

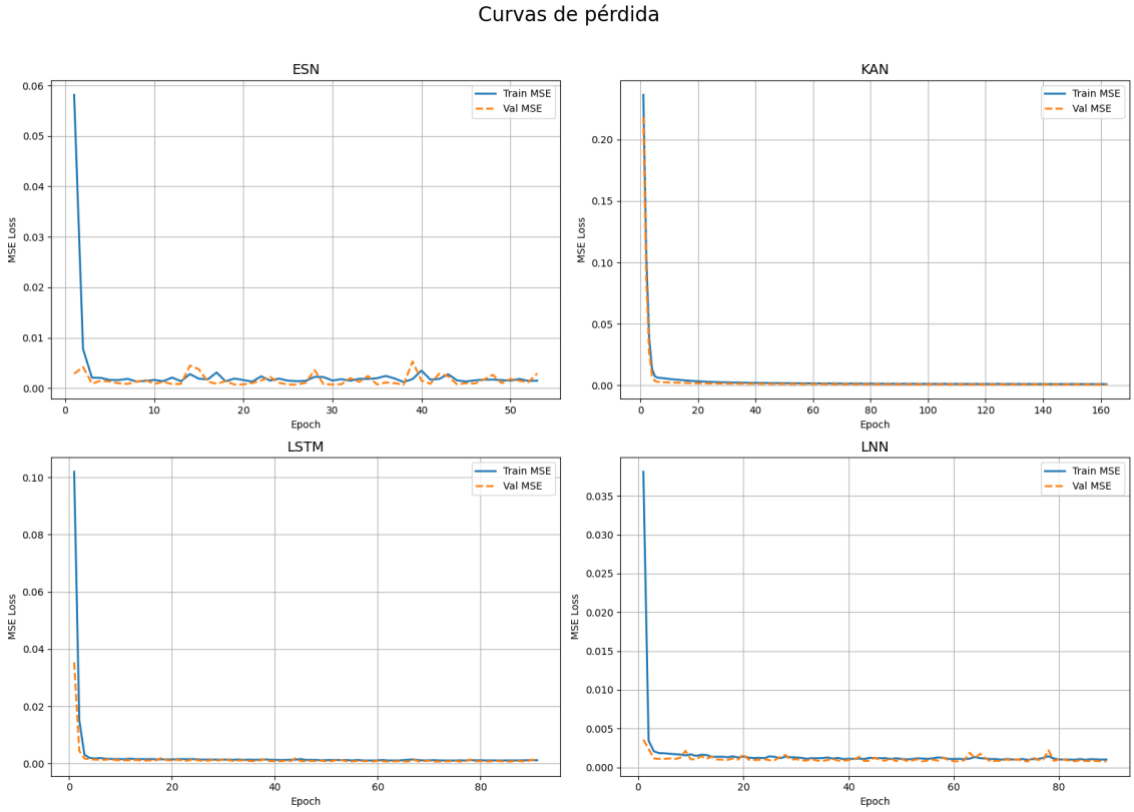

Figura 35. Curvas de pérdida de entrenamiento y validación de los modelos para Meta - Horizonte de 5 años.

Como vemos en las curvas de pérdida de la figura 35, la KAN y la LSTM descienden de forma muy limpia y sostenida, lo que indica un entrenamiento estable y sin sobreajustes; el ESN, tras la caída inicial, muestra apenas unas pequeñas ondulaciones puntuales en la MSE de validación; y la LNN, tras esa misma bajada rápida al comienzo, mantiene su línea de validación prácticamente plana, con oscilaciones ínfimas.

4.5.1.4 Coste de entrenamiento y tamaño del modelo

Modelo	Épocas	Mejor época	Tiempo de entrenamiento (s)	Tiempo a mejor época (s)	Mejor Val MSE	Parámetros
ESN	53	38	20.19s	14.55s	0.000746	2001
KAN	162	147	24.81s	22.54s	0.00074	28
LSTM	91	76	18.51s	15.47s	0.000719	71297
LNN	89	74	103.38s	86.29s	0.000754	17921

Tabla 27. Coste de entrenamiento y tamaño del modelo para Meta - Horizonte de 5 años.

Según los datos de la tabla 27, la KAN, con solo 28 parámetros y cerca de 25 s de entrenamiento, demuestra ser la opción más eficaz. El ESN maneja 2 001 parámetros en unos 20 s, el LSTM requiere casi 72 000 parámetros y entrena en torno a 18 s, y la LNN, con 17 921 parámetros, es la más costosa, necesitando más de 100 s para alcanzar su mejor validación.

4.5.2 Horizonte de 10 años

4.5.2.1 Resumen de métricas

Modelo	MSE	RMSE	MAE	R^2	MAPE (%)	SMAPE (%)
ESN	1301.74	36.08	25.60	0.95	5.23%	5.45%
KAN	180.78	13.45	9.67	0.99	2.39%	2.43%
LSTM	533.57	23.10	16.45	0.98	3.58%	3.68%
LNN	1661.07	40.76	27.82	0.93	5.54%	5.79%

Tabla 28. Resumen de métricas de modelos para Meta - Horizonte de 10 años.

Como se observa en la tabla 28, en el horizonte de diez años, la KAN sigue imbatible (R^2 0,99; MAPE 2,39 %), la LSTM mantiene un ajuste muy sólido (R^2 0,98; MAPE 3,58 %), el ESN aguanta bien pese a la ventana más amplia (R^2 0,95; MAPE 5,23 %) y la LNN se queda algo rezagada (R^2 0,93; MAPE 5,54 %).

4.5.2.2 Precio real vs predicción

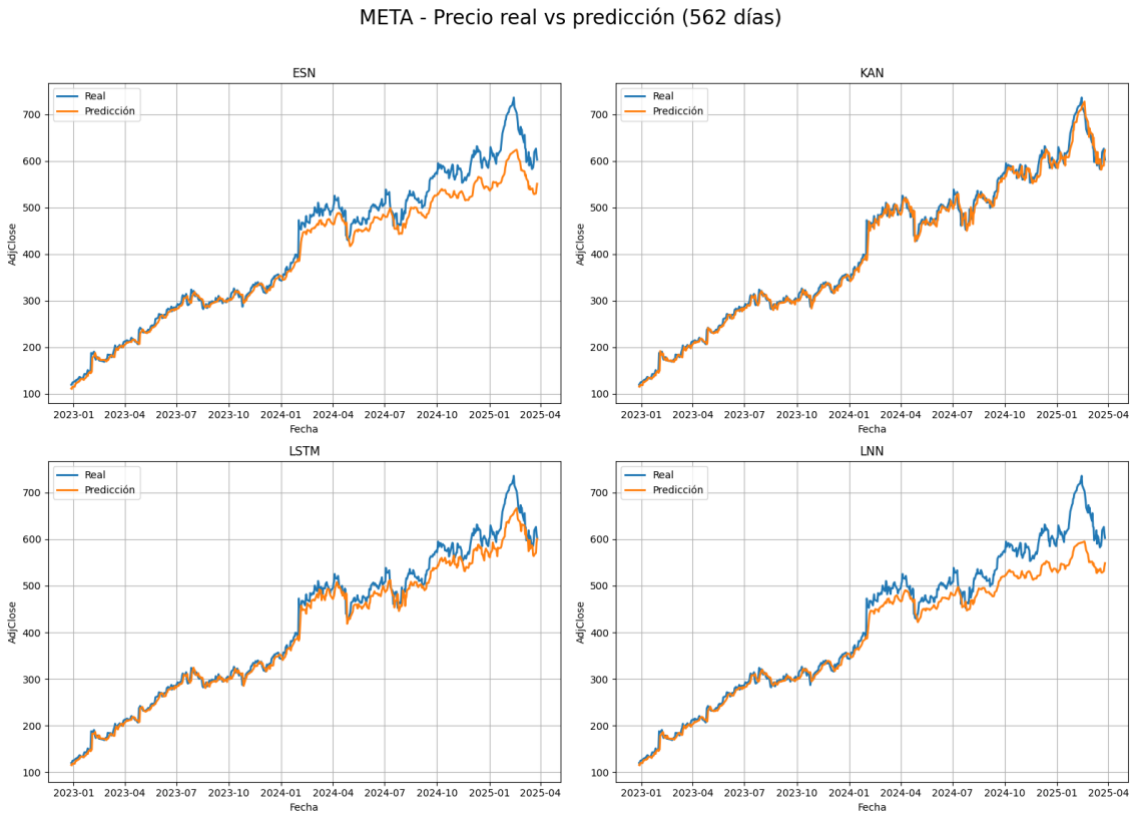

Figura 36. Precio real vs. predicción de los modelos para Meta - Horizonte de 10 años.

Tal y como se ve reflejado en la figura 36, en la comparación visual la KAN reproduce prácticamente cada punto de la serie, la LSTM acompaña muy de cerca los picos y valles, el ESN capta la tendencia, aunque se queda un poco atrás cuando la cotización acelera y la LNN ofrece una versión más suave, subestimando máximos y mínimos.

4.5.2.3 Curvas de pérdida

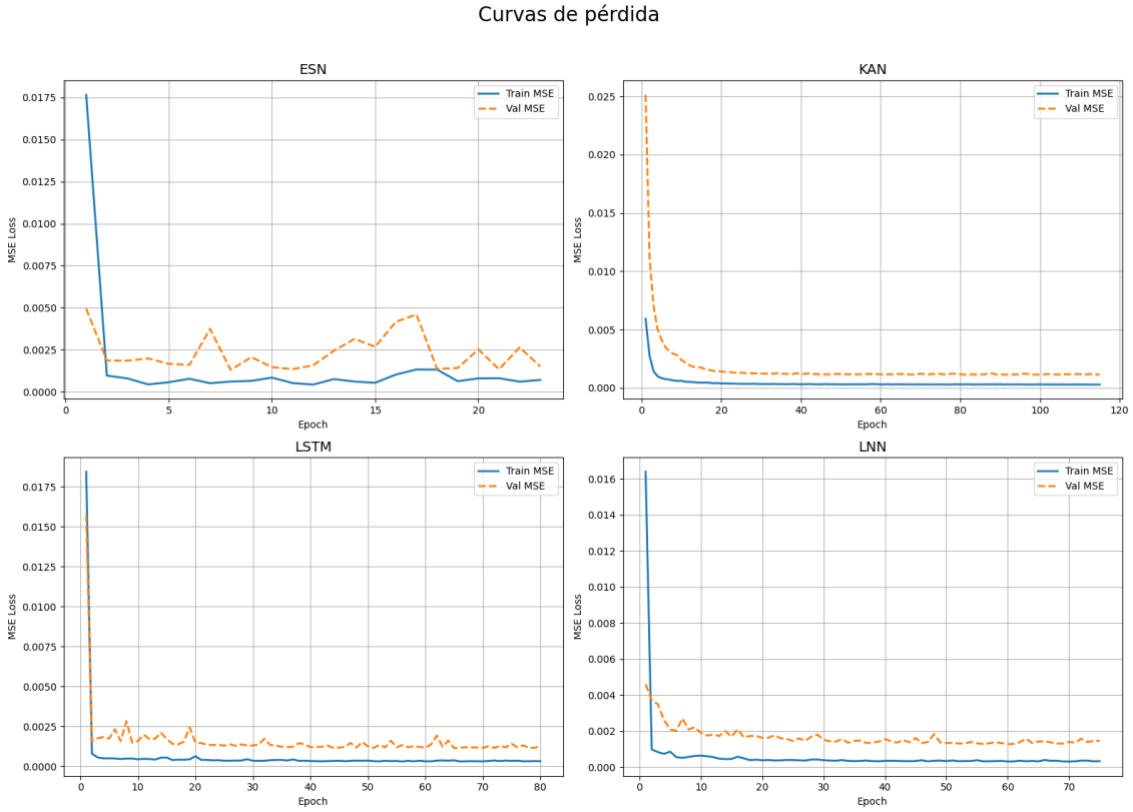

Figura 37. Curvas de pérdida de entrenamiento y validación de los modelos para Meta - Horizonte de 10 años.

Como se observa en la figura 37, las curvas de pérdida de la KAN y la LSTM caen rápido y se mantienen planas alrededor de valores mínimos; el ESN muestra ligeras ondulaciones en su validación y la LNN, una vez reducida su MSE, se mantiene muy estable sin grandes variaciones.

4.5.2.4 Coste de entrenamiento y tamaño del modelo

Modelo	Épocas	Mejor época	Tiempo de entrenamiento (s)	Tiempo a mejor época (s)	Mejor Val MSE	Parámetros
ESN	23	8	18.28s	6.52s	0.001297	2001
KAN	115	100	35.07s	30.79s	0.001145	28
LSTM	80	65	33.71s	27.30s	0.001133	71297
LNN	75	60	190.85s	152.93s	0.001268	17921

Tabla 29. Coste de entrenamiento y tamaño del modelo para Meta - Horizonte de 10 años.

Según los datos de la tabla 29, en cuanto a recursos, la KAN entrena en unos 35 s con solo 28 parámetros; el ESN, en 18 s con 2 001 parámetros; la LSTM, en 34 s con 71 297 parámetros; y la LNN, la más costosa, tarda cerca de 191 s manejando 17 921 parámetros.

4.5.3 Horizonte máximo

4.5.3.1 Resumen de métricas

Modelo	MSE	RMSE	MAE	R^2	MAPE (%)	SMAPE (%)
ESN	3069.47	55.40	36.84	0.90	8.55%	8.91%
KAN	154.88	12.45	8.72	0.99	2.80%	2.80%
LSTM	922.22	30.37	20.21	0.97	5.04%	5.16%
LNN	1795.04	42.37	27.74	0.94	6.28%	6.51%

Tabla 30. Resumen de métricas de modelos para Meta - Horizonte máximo.

Según los datos de la tabla 30, en este horizonte, la KAN sigue dominando con un R^2 de 0,99 y un MAPE de apenas 2,80 %, ofreciendo una reproducción casi exacta de la dinámica histórica. La LSTM obtiene un ajuste muy sólido (R^2=0,97; MAPE 5,04 %), mientras que la LNN, pese a manejar mejor que el ESN los baches de la serie, registra un error significativamente mayor (MAPE 6,28 %; R^2=0,94). El ESN cierra el grupo con un rendimiento más modesto (MAPE 8,55 %; R^2=0,90), evidenciando sus limitaciones al extender tanto la ventana de predicción.

4.5.3.2 Precio real vs predicción

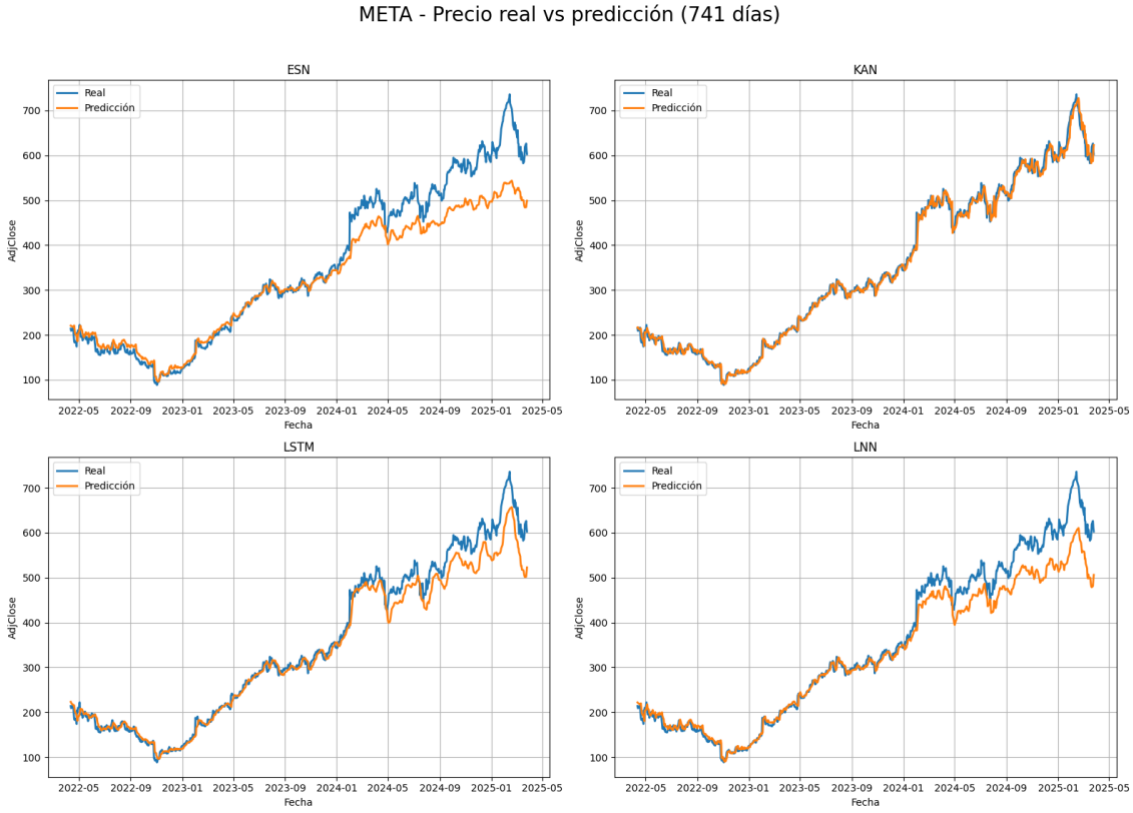

Figura 38. Precio real vs. predicción de los modelos para Meta - Horizonte máximo.

Como se observa en la figura 38, la comparación visual confirma estos números: la curva naranja de la KAN prácticamente solapa la azul original durante todo el período, incluso en los picos de inicios de 2025. La LSTM también sigue con muy poco desfase las subidas y bajadas, aunque suaviza ligeramente los extremos. El ESN capta bien la tendencia general pero se sitúa por debajo en los momentos más volátiles. La LNN exhibe un trazado moderado, fiel a la pauta, pero sistemáticamente contenido en sus valores máximos y mínimos.

4.5.3.3 Curvas de pérdida

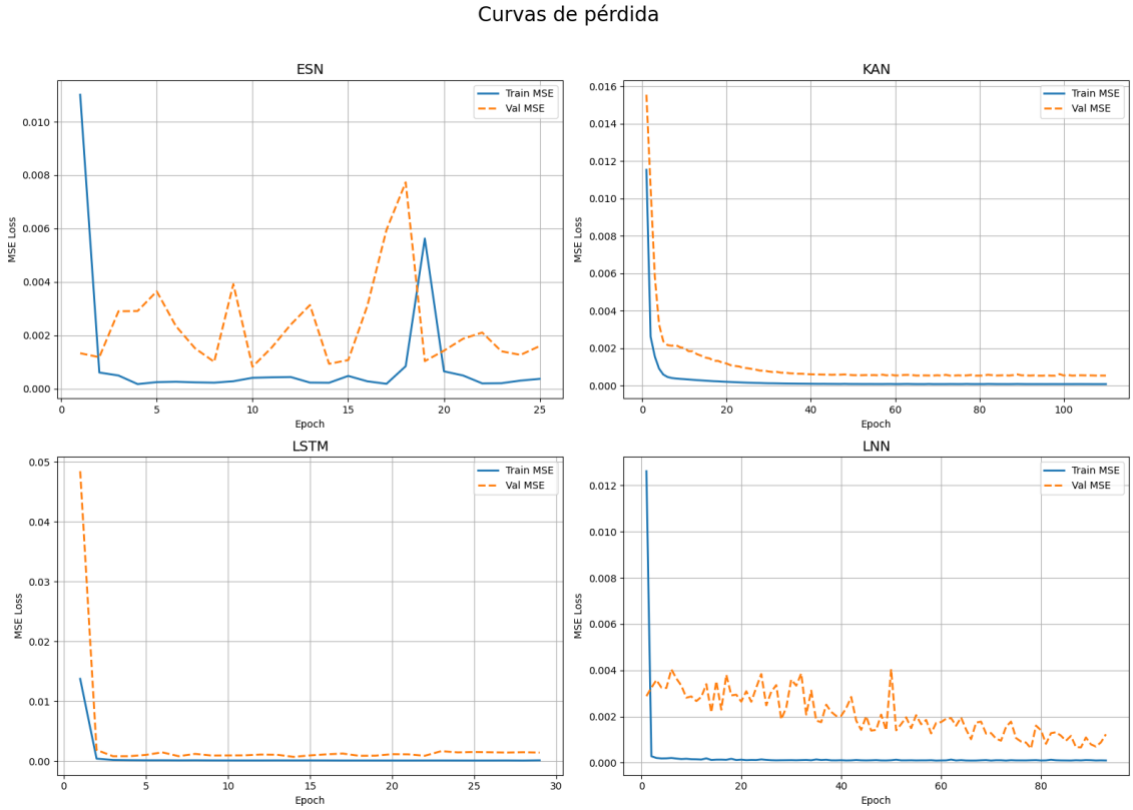

Figura 39. Curvas de pérdida de entrenamiento y validación de los modelos para Meta - Horizonte máximo.

Como muestra la figura 39, las trayectorias de KAN y LSTM muestran una caída rápida de la MSE de validación y luego se mantienen planas en torno a valores mínimos, señal de un entrenamiento estable. El ESN presenta pequeñas ondulaciones en su pérdida de validación, reflejo de altibajos puntuales en el ajuste. La LNN, tras reducir su error inicial, conserva una MSE muy baja con ligeras oscilaciones decrecientes, sin picos bruscos.

4.5.3.4 Coste de entrenamiento y tamaño del modelo

Modelo	Épocas	Mejor época	Tiempo de entrenamiento (s)	Tiempo a mejor época (s)	Mejor Val MSE	Parámetros
ESN	25	10	23.81s	9.68s	0.000825	2001
KAN	110	95	44.10s	38.41s	0.000534	28
LSTM	29	14	15.34s	7.23s	0.000717	71297
LNN	93	78	288.65s	241.87s	0.000614	17921

Tabla 31. Coste de entrenamiento y tamaño del modelo para Meta - Horizonte máximo.

Tal y como se muestra en la tabla 31, la KAN completa su entrenamiento en unos 44 s manejando únicamente 28 parámetros, lo que la convierte en la opción más ágil. El ESN tarda alrededor de 24 s con 2 001 parámetros, un compromiso razonable. La LSTM exige unos 15 s pero con casi 71 300 parámetros, y la LNN, con 17 921 parámetros, es con diferencia la más costosa en tiempo (cerca de 289 s) para alcanzar su mínimo de validación.

4.6 Tesla (TSLA)

4.6.1 Horizonte de 5 años

4.6.1.1 Resumen de métricas

Modelo	MSE	RMSE	MAE	R^2	MAPE (%)	SMAPE (%)
ESN	341.27	18.47	13.79	0.95	5.33%	5.28%
KAN	288.01	16.97	12.26	0.96	4.53%	4.56%
LSTM	324.80	18.02	12.70	0.95	4.63%	4.68%
LNN	330.63	18.18	13.07	0.95	4.83%	4.89%

Tabla 32. Resumen de métricas de modelos para Tesla - Horizonte de 5 años.

Según los datos de la tabla 32, la KAN encabeza el grupo con un MAPE del 4,53 % y R^2 = 0,96. Muy cerca quedan la LSTM (MAPE 4,63 %, R^2 = 0,95) y la LNN (MAPE 4,83 %, R^2 = 0,95). El ESN es el menos preciso dentro de un rango aún aceptable (MAPE 5,33 %, R^2 = 0,95).

4.6.1.2 Precio real vs predicción

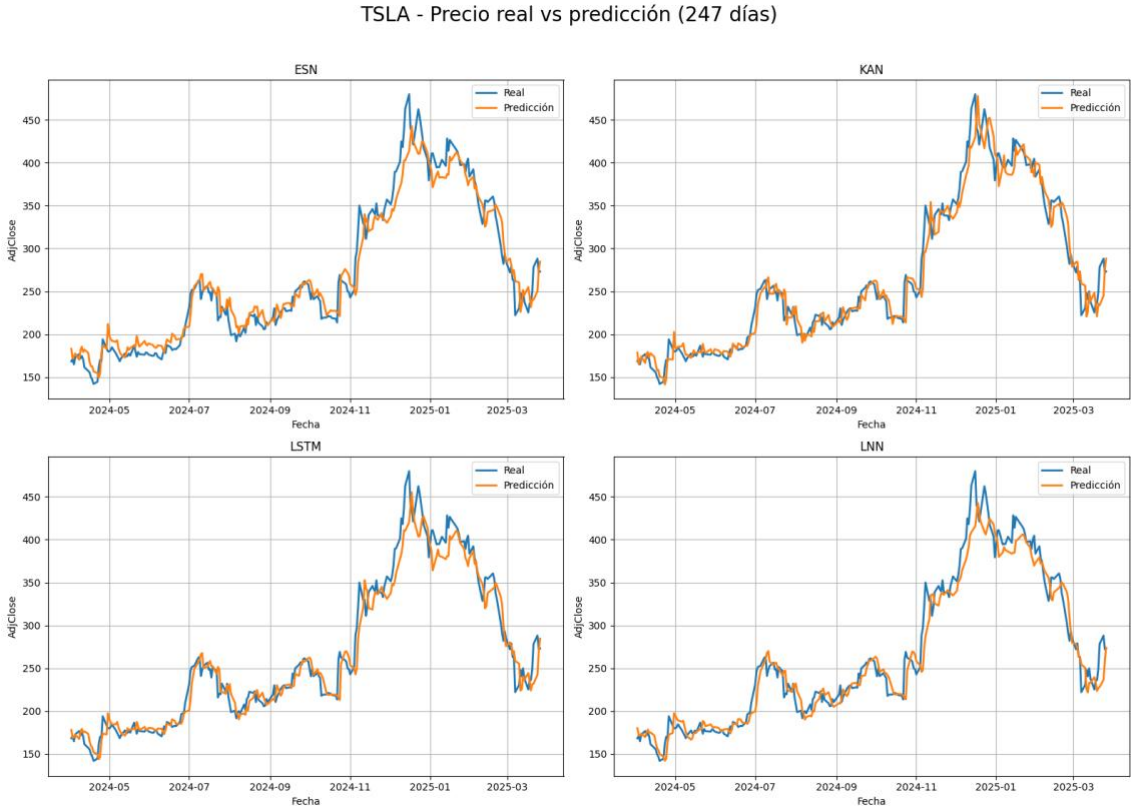

Figura 40. Precio real vs. predicción de los modelos para Tesla - Horizonte de 5 años.

Como se observa en la figura 40, en los 247 días de test para Tesla, la KAN se ajusta casi punto a punto al movimiento real, capturando tanto los saltos de fin de año como la corrección posterior. La LSTM sigue de cerca esas oscilaciones, aunque suaviza ligeramente los picos máximos. La LNN también refleja con fidelidad la pauta general, si bien tiende a quedarse un poco por debajo en los mínimos de marzo de 2025. El ESN, por su parte, muestra la misma tendencia, pero con un sesgo a subestimar los valores extremos, especialmente en los ascensos más pronunciados de finales de 2024.

4.6.1.3 Curvas de pérdida

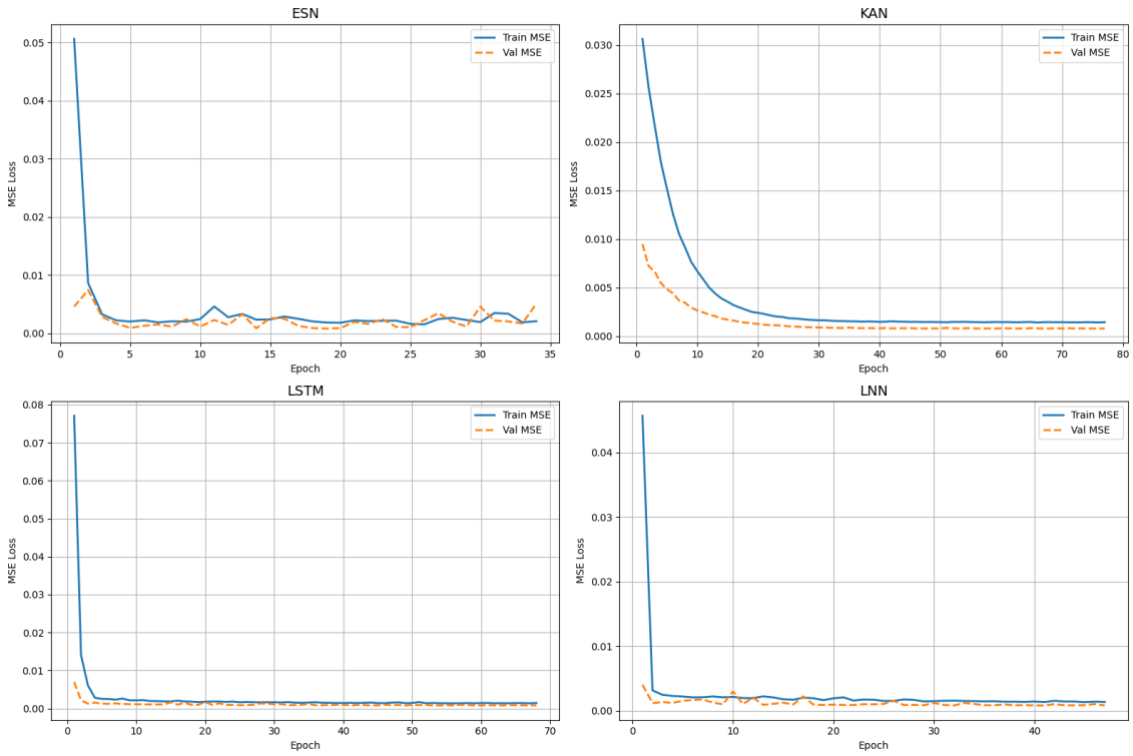

Figura 41. Curvas de pérdida de entrenamiento y validación de los modelos para Tesla - Horizonte de 5 años.

Tal y como se muestra en la figura 41, KAN y LSTM presentan una caída muy limpia de la MSE de validación hasta estabilizarse en un nivel bajo, indicativo de un entrenamiento sólido y sin sobreajuste. El ESN, tras reducir su error inicial, exhibe pequeñas ondulaciones en la MSE de validación, muestra de altibajos puntuales en el ajuste. La LNN logra también una pérdida muy baja y prácticamente plana, con ligeras variaciones menores a lo largo del entrenamiento, lo que refleja su capacidad para fijar rápidamente el error y mantenerlo estable.

4.6.1.4 Coste de entrenamiento y tamaño del modelo

Modelo	Épocas	Mejor época	Tiempo de entrenamiento (s)	Tiempo a mejor época (s)	Mejor Val MSE	Parámetros
ESN	34	19	14.06s	7.81s	0.000801	2001
KAN	77	62	13.23s	10.78s	0.000765	28
LSTM	68	53	14.02s	11.01s	0.000777	71297
LNN	47	32	56.43s	37.66s	0.000804	17921

Tabla 33. Coste de entrenamiento y tamaño del modelo para Tesla - Horizonte de 5 años.

Según los datos de la tabla 33, la KAN ajusta 28 parámetros en unos 13 s; el ESN emplea 14 s para 2 001 parámetros. La LSTM necesita unos 14 s pero mueve 71 297 pesos. La LNN es la más costosa: cerca de 56 s de cómputo y 17 921 parámetros para lograr su mejor validación.

4.6.2 Horizonte de 10 años

4.6.2.1 Resumen de métricas

Modelo	MSE	RMSE	MAE	R^2	MAPE (%)	SMAPE (%)
ESN	318.26	17.84	13.88	0.93	6.06%	5.92%
KAN	179.35	13.39	9.58	0.96	4.00%	4.03%
LSTM	205.27	14.33	10.31	0.96	4.31%	4.30%
LNN	378.19	19.45	14.79	0.92	6.26%	6.21%

Tabla 34. Resumen de métricas de modelos para Tesla - Horizonte de 10 años.

Como se muestra en la tabla 34, la KAN conserva la primera posición (MAPE 4,00 %; R^2 0,96). La LSTM queda muy cerca (MAPE 4,31 %; R^2 0,96). El ESN baja algo la precisión (MAPE 6,06 %; R^2 0,93) y la LNN cierra el grupo con el mayor error (MAPE 6,26 %; R^2 0,92).

4.6.2.2 Precio real vs predicción

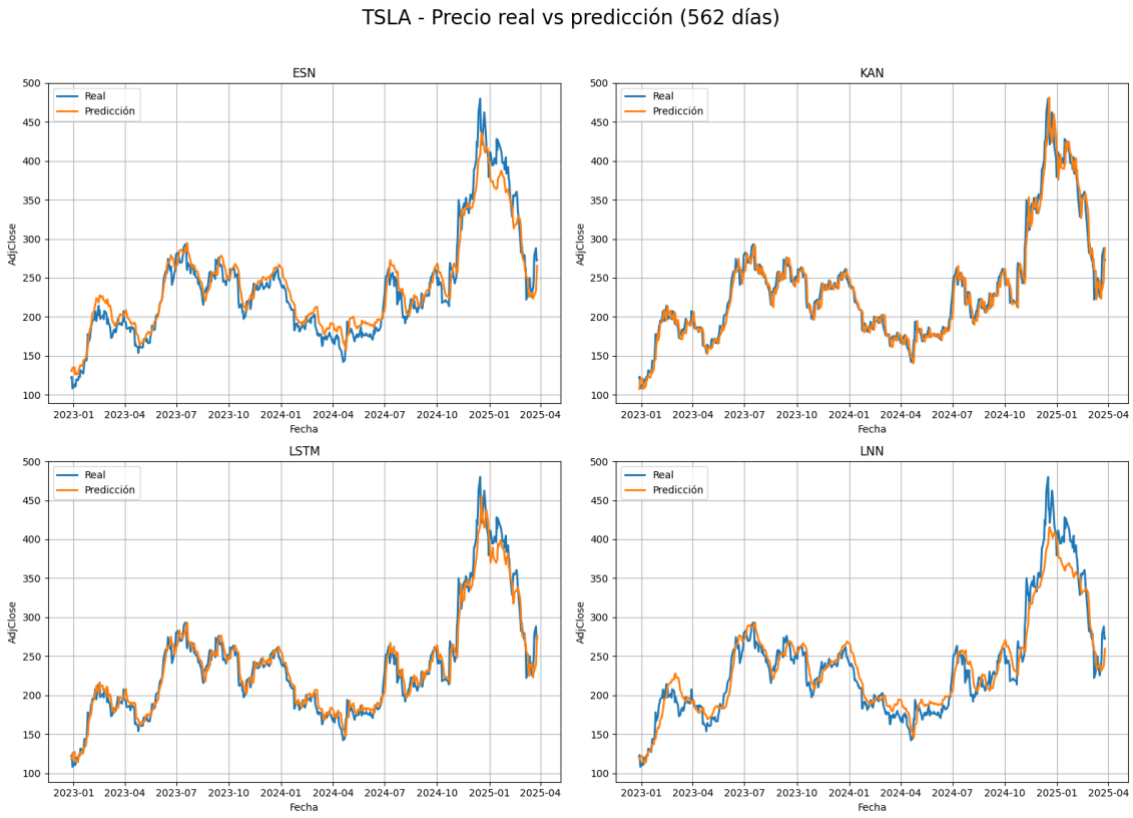

Figura 42. Precio real vs. predicción de los modelos para Tesla - Horizonte de 10 años.

Tal y como se observa en la figura 42, los 562 días de test las líneas real y predicha de la KAN casi se solapan, incluso en el pico de finales de 2024 y la caída posterior. La LSTM reproduce bien esos movimientos aunque suaviza levemente los máximos. El ESN sigue la tendencia general, pero se queda corto en los puntos más altos, mientras que la LNN mantiene la forma global con cierta sub-estimación tanto en los picos como en los valles.

4.6.2.3 Curvas de pérdida

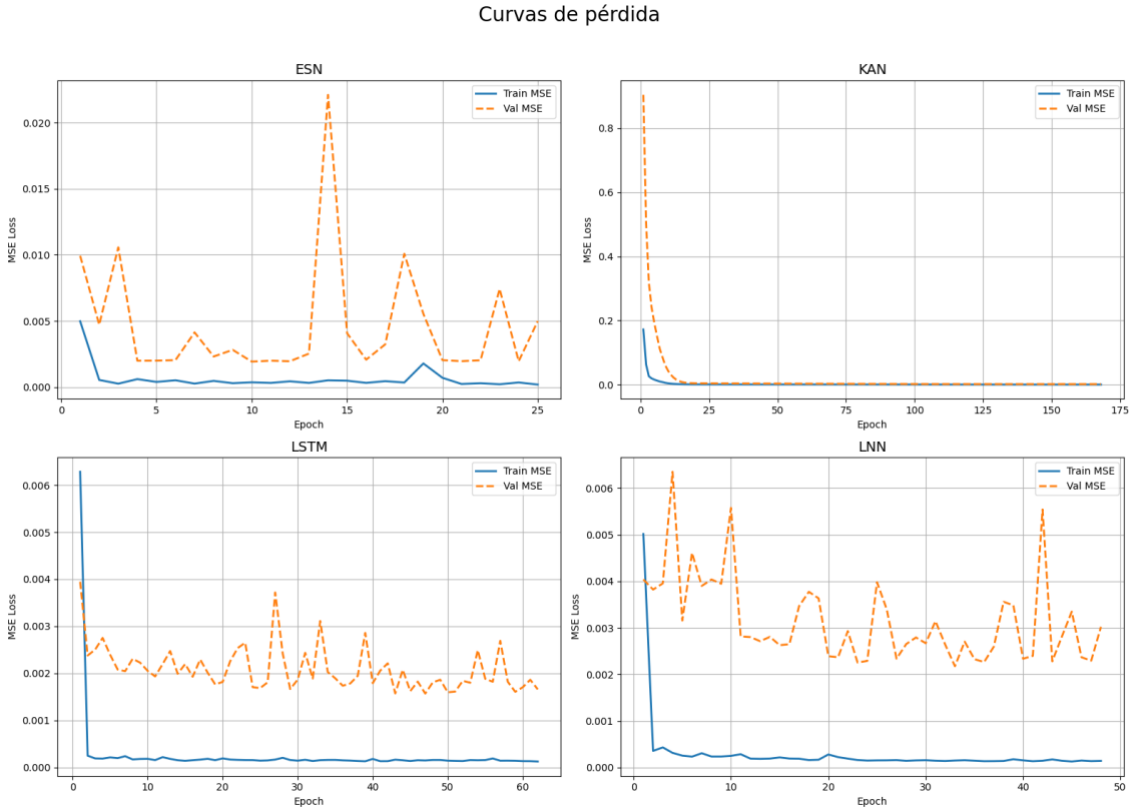

Figura 43. Curvas de pérdida de entrenamiento y validación de los modelos para Tesla - Horizonte de 10 años.

Como se observa en la figura 43, la KAN reduce su MSE de validación con rapidez y se estabiliza en valores muy bajos. Las demás no llegan a reducir mucho su MSE y presentan oscilaciones bastante pronunciadas.

4.6.2.4 Coste de entrenamiento y tamaño del modelo

Modelo	Épocas	Mejor época	Tiempo de entrenamiento (s)	Tiempo a mejor época (s)	Mejor Val MSE	Parámetros
ESN	25	10	20.14s	8.35s	0.001924	2001
KAN	168	153	53.05s	48.27s	0.001467	28
LSTM	62	47	26.02s	19.72s	0.001568	71297
LNN	48	33	116.51s	80.71s	0.002175	17921

Tabla 35. Coste de entrenamiento y tamaño del modelo para Tesla - Horizonte de 10 años.

Según los datos de la tabla 35, la KAN (28 parámetros) completa 168 épocas en unos 53 s, alcanzando su mejor punto poco antes del minuto. El ESN (2 001 parámetros) necesita solo 20 s para terminar y fija su mínimo de validación a mitad de carrera. La LSTM emplea 26 s para 62 épocas y maneja 71 297 pesos. La LNN es la más lenta y pesada: 48 épocas, casi 2 minutos de cómputo y 17 921 parámetros.

4.6.3 Horizonte máximo

4.6.3.1 Resumen de métricas

Modelo	MSE	RMSE	MAE	R^2	MAPE (%)	SMAPE (%)
ESN	2358.74	48.57	34.31	0.44	12.66%	13.84%
KAN	311.47	17.65	13.16	0.93	5.59%	5.56%
LSTM	359.88	18.97	14.13	0.91	5.91%	6.08%
LNN	298.31	17.27	12.18	0.93	4.88%	4.97%

Tabla 36. Resumen de métricas de modelos para Tesla - Horizonte máximo.

Según los datos de la tabla 36, la KAN mantiene la menor desviación (MAPE 5,59 %, R^2 0,93), la LNN se le acerca (4,88 %; R^2 0,93) y la LSTM queda apenas un punto por encima (5,91 %; R^2 0,91). El ESN se rezaga con un MAPE superior al 12 % y un R^2 de solo 0,44, señal de que pierde amplitud en los movimientos más bruscos.

4.6.3.2 Precio real vs predicción

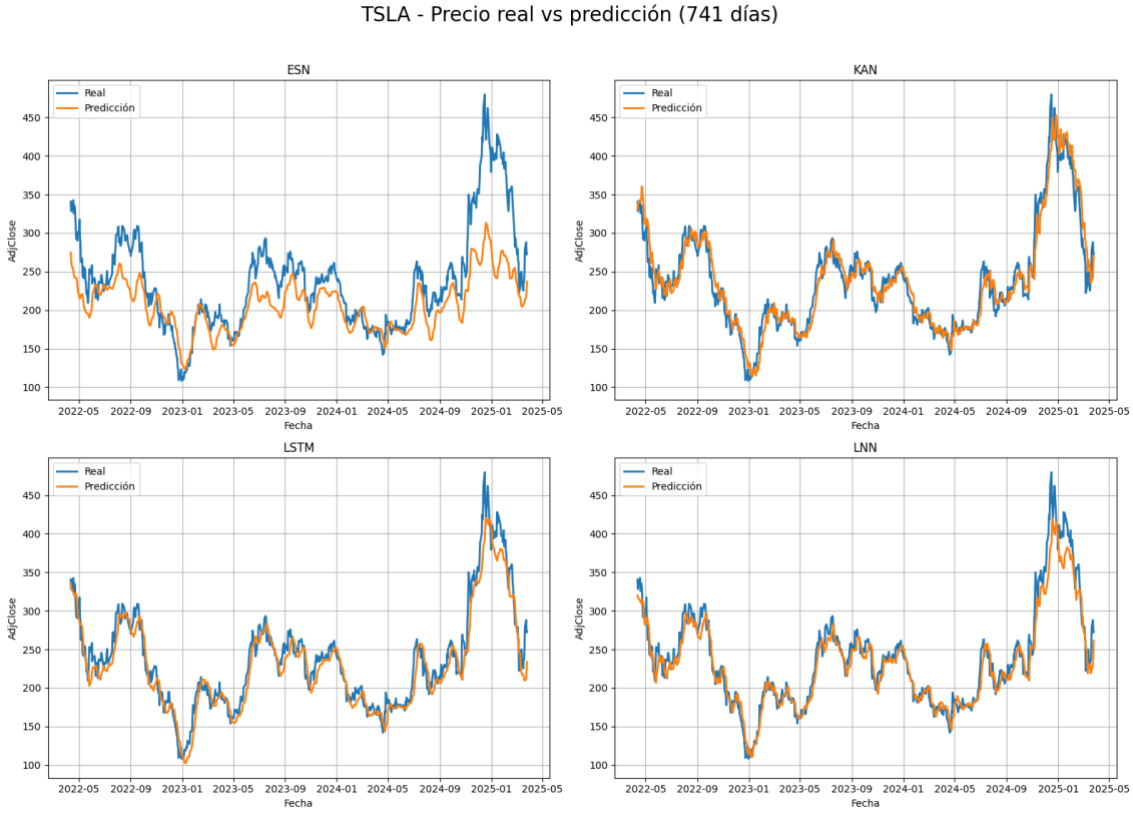

Figura 44. Precio real vs. predicción de los modelos para Tesla - Horizonte máximo.

Tal y como se observa en los 741 días de test de la figura 44, la KAN sigue con detalle los ascensos de otoño 2024 y el fuerte valle de enero 2025. La LNN reproduce de forma bastante fiel ambas direcciones, pero recorta algo los extremos. La LSTM acompaña la tendencia general con un ligero suavizado tanto en picos como en valles. El ESN, en cambio, muestra la pauta global pero subestima de forma clara el máximo absoluto y la caída posterior.

4.6.3.3 Curvas de pérdida

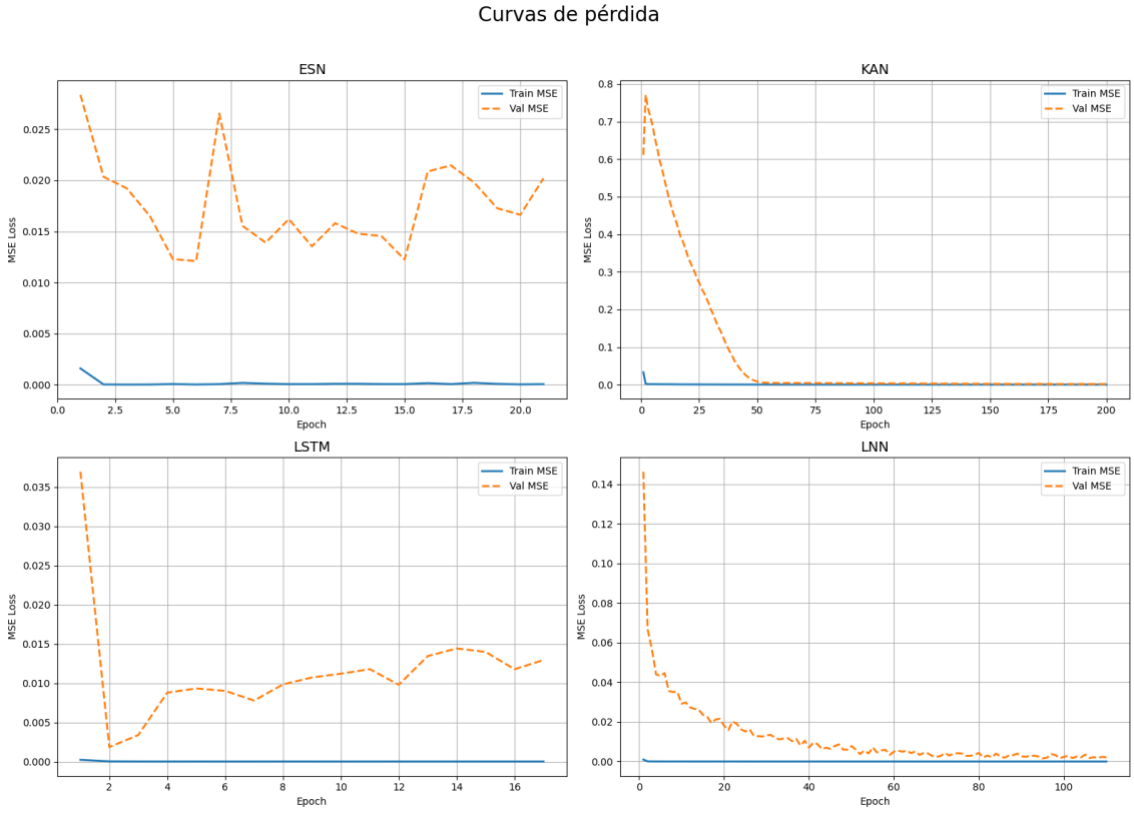

Figura 45. Curvas de pérdida de entrenamiento y validación de los modelos para Tesla - Horizonte máximo.

Según los gráficos de la figura 45, la curva de la KAN desciende rápido y se estabiliza en valores muy bajos sin sobresaltos; la LNN tarda algo más en asentarse, aunque lo hace de forma progresiva y estable. La LSTM reduce pronto la MSE, pero su validación presenta varios repuntes moderados. El ESN es el más inestable: la validación oscila durante casi todo el entrenamiento y nunca alcanza un nivel tan bajo como el resto.

4.6.3.4 Coste de entrenamiento y tamaño del modelo

Modelo	Épocas	Mejor época	Tiempo de entrenamiento (s)	Tiempo a mejor época (s)	Mejor Val MSE	Parámetros
ESN	21	6	20.09s	5.74s	0.0121	2001
KAN	200	199	80.95s	80.57s	0.001294	28
LSTM	17	2	9.26s	1.14s	0.001847	71297
LNN	110	95	334.97s	290.28s	0.00157	17921

Tabla 37. Coste de entrenamiento y tamaño del modelo para Tesla - Horizonte máximo.

Según los datos de la tabla 37, el ESN es el más rápido (≈ 20 s) con 2 001 parámetros, aunque su ajuste final es pobre. La KAN emplea unos 81 s para afinar sus 28 parámetros, ofreciendo el mejor equilibrio entre precisión y complejidad. La LSTM entrena en 9 s pese a sus 71 k pesos, mientras que la LNN necesita más de cinco minutos para converger con 17 921 parámetros.

4.7 Google (GOOGL)

4.7.1 Horizonte de 5 años

4.7.1.1 Resumen de métricas

Modelo	MSE	RMSE	MAE	R^2	MAPE (%)	SMAPE (%)
ESN	122.82	11.08	9.82	0.23	5.55%	5.75%
KAN	22.91	4.79	3.55	0.86	2.05%	2.05%
LSTM	39.95	6.32	5.21	0.75	2.96%	3.01%
LNN	109.31	10.46	9.28	0.31	5.26%	5.44%

Tabla 38. Resumen de métricas de modelos para Google - Horizonte de 5 años.

Tal y como se muestra en la tabla 38, la KAN obtiene el mejor ajuste (R^2 0.86; MAPE 2.05 %). La LSTM queda segunda (R^2 0.75; MAPE 2.96 %). La LNN y el ESN se alejan bastante: ambas superan el 5 % de error y su R^2 no llega a 0.32 (ESN 0.23; LNN 0.31).

4.7.1.2 Precio real vs predicción

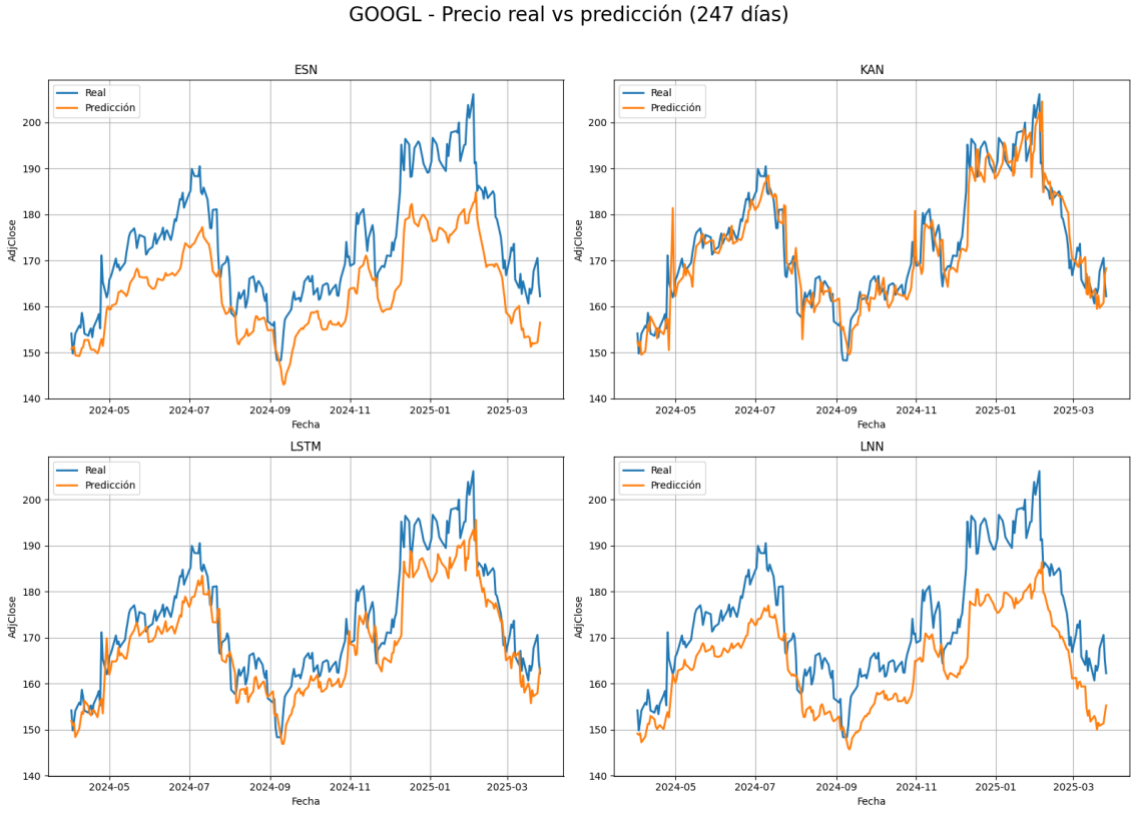

Figura 46. Precio real vs. predicción de los modelos para Google - Horizonte de 5 años.

Como se observa en la figura 46, la KAN sigue de cerca cada subida y cada retroceso, incluidos los picos de enero-febrero 2025. La LSTM acompaña bien la tendencia aunque rebaja ligeramente los extremos. El ESN capta las direcciones pero subestima la mayoría de máximos, y la LNN traza una versión algo "aplanada", por debajo de los valles y de los picos más pronunciados.

4.7.1.3 Curvas de pérdida

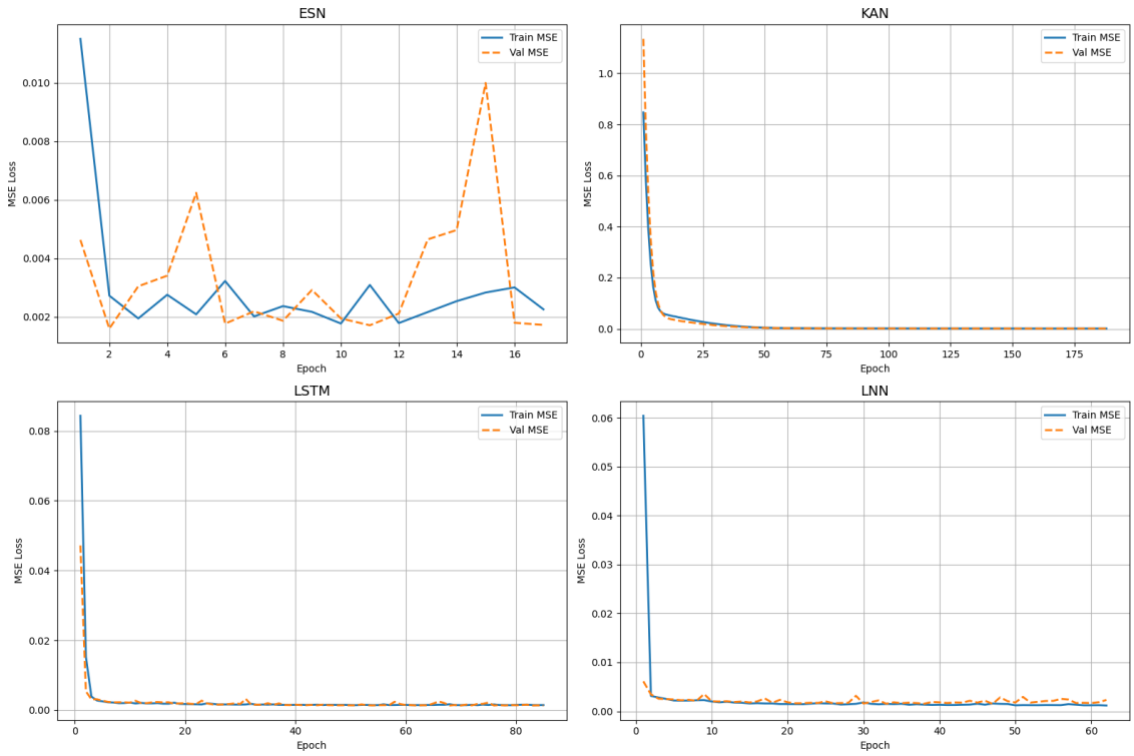

Figura 47. Curvas de pérdida de entrenamiento y validación de los modelos para Google - Horizonte de 5 años.

Como se observa en la figura 47, KAN desciende rápido a una MSE de validación muy baja y estable. LSTM converge con un perfil parecido, mostrando solo pequeñas oscilaciones. El ESN mantiene la validación claramente más alta y con altibajos continuos; la LNN baja pronto el error, pero conserva ligeras ondulaciones a lo largo del entrenamiento.

4.7.1.4 Coste de entrenamiento y tamaño del modelo

Modelo	Épocas	Mejor época	Tiempo de entrenamiento (s)	Tiempo a mejor época (s)	Mejor Val MSE	Parámetros
ESN	17	2	6.65s	0.79s	0.001605	2001
KAN	188	173	26.17s	24.08s	0.001331	28
LSTM	85	70	17.19s	14.13s	0.001305	71297
LNN	62	47	71.18s	53.94s	0.001511	17921

Tabla 39. Coste de entrenamiento y tamaño del modelo para Google - Horizonte de 5 años.

Como se muestra en la tabla 39, ESN es la más rápida (≈ 7 s totales, mejor modelo en menos de 1 s) con 2 001 parámetros. La KAN tarda unos 26 s —pero sigue siendo muy ligera, solo 28 parámetros—. La LSTM emplea 17 s para ajustar 71 297 pesos, y la LNN necesita cerca de 54 s para sus 17 921 parámetros.

4.7.2 Horizonte de 10 años

4.7.2.1 Resumen de métricas

Modelo	MSE	RMSE	MAE	R^2	MAPE (%)	SMAPE (%)
ESN	90.52	9.51	7.22	0.89	4.51%	4.65%
KAN	38.82	6.23	4.95	0.95	3.38%	3.42%
LSTM	20.42	4.52	3.52	0.98	2.42%	2.44%
LNN	33.23	5.76	4.37	0.96	2.88%	2.93%

Tabla 40. Resumen de métricas de modelos para Google - Horizonte de 10 años.

Según los datos de la tabla 40, LSTM es el modelo más preciso (MAPE 2,42 %, R^2 0,98). Le sigue la KAN (MAPE 3,38 %, R^2 0,95). La LNN conserva un error moderado (MAPE 2,88 %, R^2 0,96). El ESN, aunque aceptable, queda el último (MAPE 4,51 %, R^2 0,89).

4.7.2.2 Precio real vs predicción

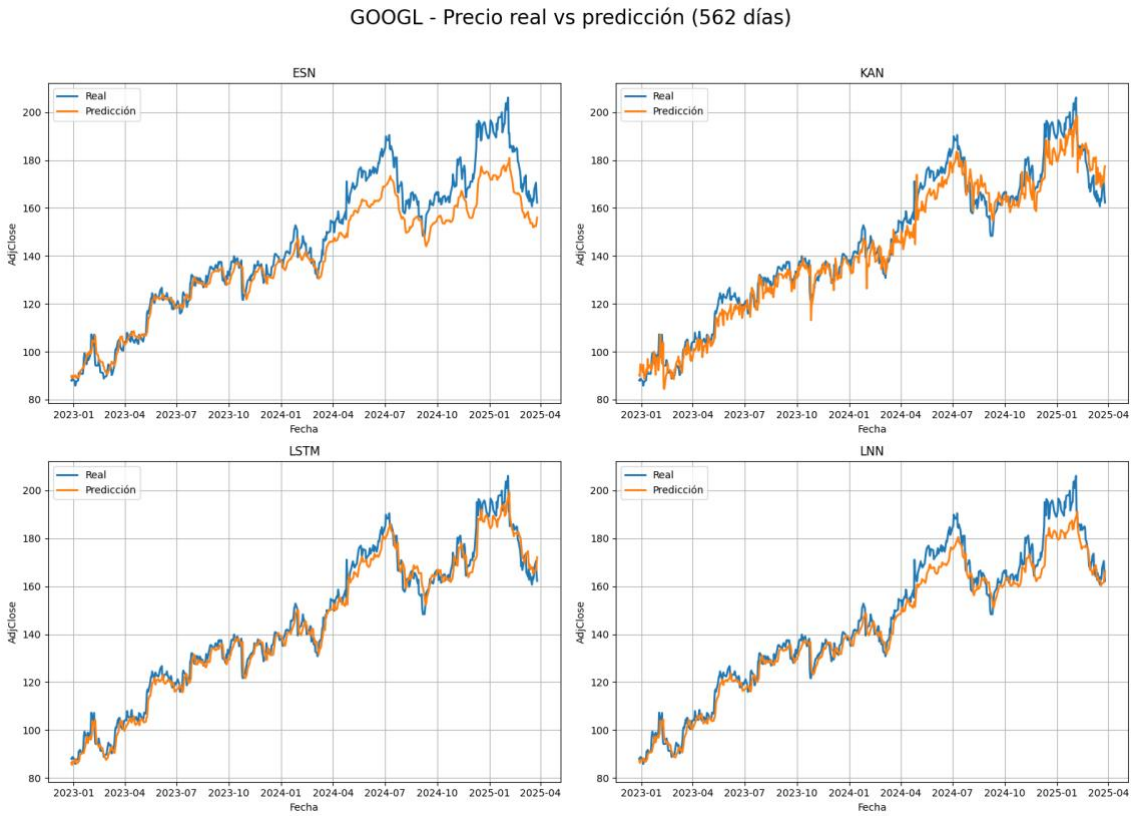

Figura 48. Precio real vs. predicción de los modelos para Google - Horizonte de 10 años.

Como se observa en la figura 48, en los 562 días de test la KAN y la LSTM casi calcaban la serie real; la LSTM ajusta algo mejor los picos y los valles. El ESN sigue la tendencia principal, pero subestima los máximos, y la LNN acompaña la forma general con cierto amortiguamiento de los extremos.

4.7.2.3 Curvas de pérdida

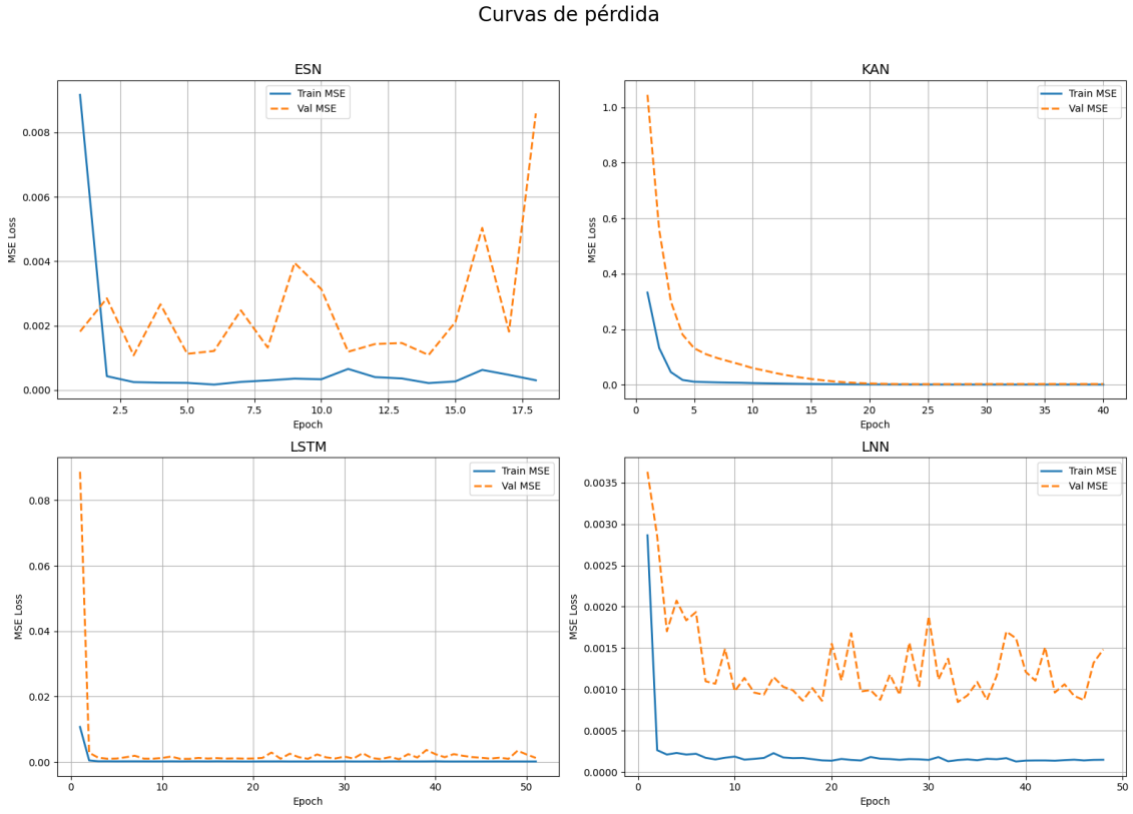

Figura 49. Curvas de pérdida de entrenamiento y validación de los modelos para Google - Horizonte de 10 años.

Como se observa en la figura 49, KAN muestra un descenso rápido y limpio de la MSE de validación hasta valores muy bajos. LSTM converge con un perfil igualmente estable. El ESN presenta oscilaciones regulares en la validación, signo de ajuste menos consistente. La LNN baja pronto el error y luego mantiene una línea plana con pequeñas ondulaciones.

4.7.2.4 Coste de entrenamiento y tamaño del modelo

Modelo	Épocas	Mejor época	Tiempo de entrenamiento (s)	Tiempo a mejor época (s)	Mejor Val MSE	Parámetros
ESN	18	3	13.22s	2.24s	0.001079	2001
KAN	40	25	12.64s	7.93s	0.001723	28
LSTM	51	36	20.87s	14.77s	0.000826	71297
LNN	48	33	114.65s	79.82s	0.000846	17921

Tabla 41. Coste de entrenamiento y tamaño del modelo para Google - Horizonte de 10 años.

Según los datos de la tabla 41, ESN entrenó en 13 s y fijó su mejor punto al inicio, moviendo 2 001 parámetros. La KAN tardó unos 13 s en total con solo 28 parámetros. La LSTM necesitó 21 s para ajustar 71 297 pesos, mientras que la LNN fue la más lenta: casi 115 s para 17 921 parámetros.

4.7.3 Horizonte máximo

4.7.3.1 Resumen de métricas

Modelo	MSE	RMSE	MAE	R^2	MAPE (%)	SMAPE (%)
ESN	525.50	22.92	18.52	0.45	12.29%	13.41%
KAN	14.71	3.84	2.90	0.98	2.20%	2.20%
LSTM	29.08	5.39	4.18	0.97	3.04%	3.06%
LNN	53.69	7.33	5.67	0.94	3.85%	3.94%

Tabla 42. Resumen de métricas de modelos para Google - Horizonte máximo.

Tal y como se muestra en la tabla 42, la KAN es la más precisa (MAPE 2,20 %, R^2 0,98). Le siguen la LSTM (3,04 %, 0,97) y la LNN (3,85 %, 0,94). El ESN se queda muy atrás con un error que supera el 12 % y un R^2 de sólo 0,45.

4.7.3.2 Precio real vs predicción

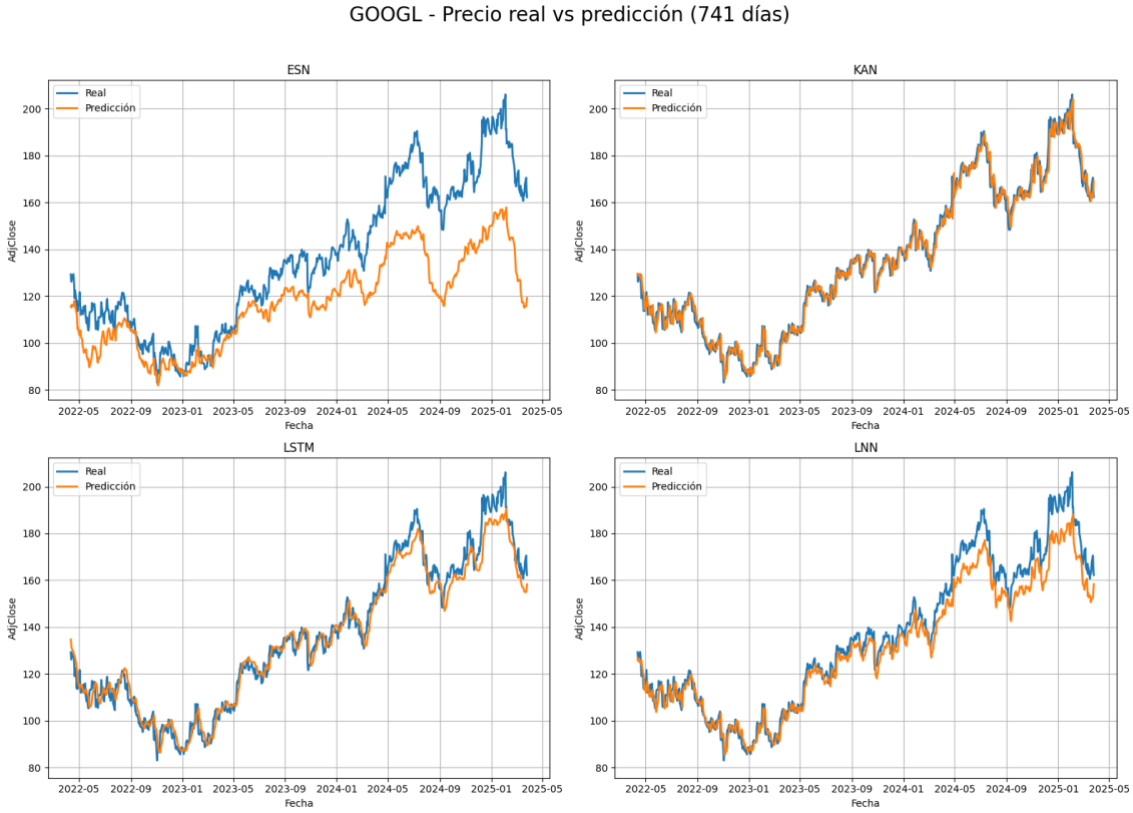

Figura 50. Precio real vs. predicción de los modelos para Google - Horizonte máximo.

Según el gráfico de la figura 50, en los 741 días de prueba la KAN prácticamente calca la serie real, incluidas las subidas sostenidas de 2024 y el giro bajista de marzo 2025. La LSTM acompaña bien la forma general, aunque suaviza algo los extremos. La LNN reproduce la tendencia, pero con menor amplitud en picos y valles. El ESN sigue la dirección global, aunque subestima de forma clara los máximos y mínimos más acusados.

4.7.3.3 Curvas de pérdida

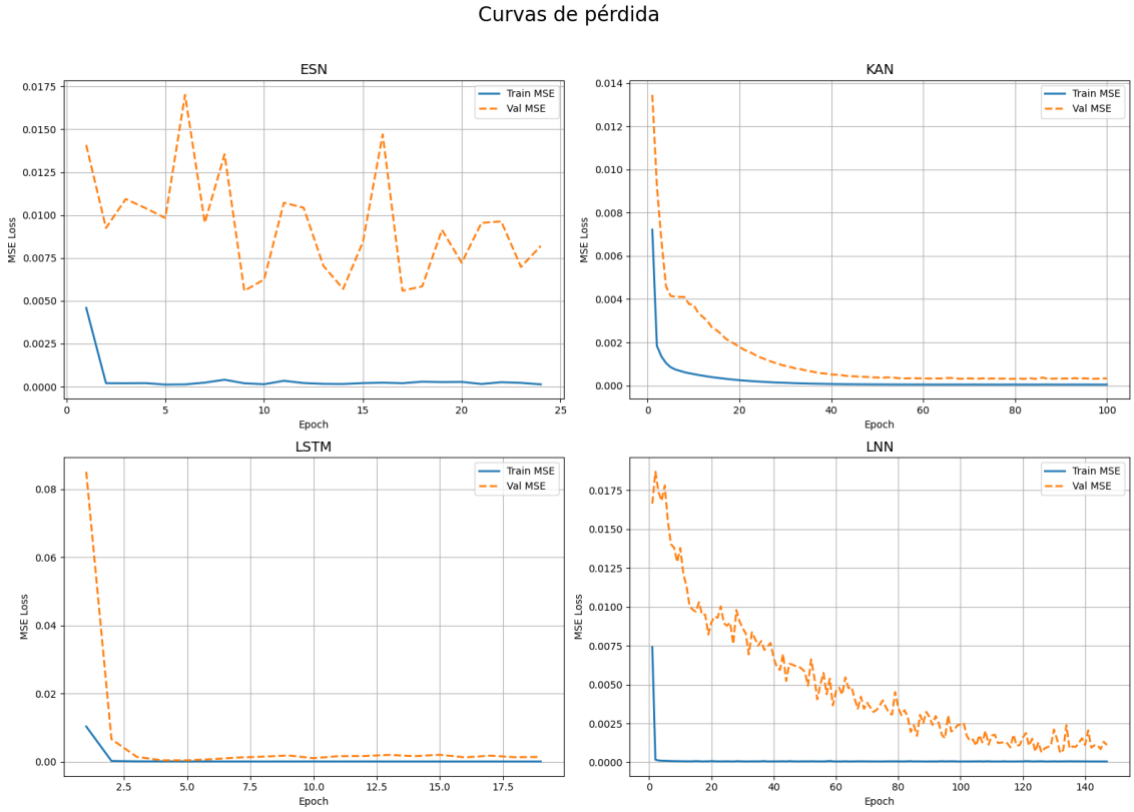

Figura 51. Curvas de pérdida de entrenamiento y validación de los modelos para Google - Horizonte máximo.

Según los gráficos de la figura 51, la KAN baja su MSE de validación con rapidez y se estabiliza en valores muy reducidos. La LSTM muestra un patrón similar, con una ligera meseta final. El ESN mantiene la validación en un nivel mucho más alto y salta de forma irregular durante todo el entrenamiento. La LNN parte de un error elevado y lo va reduciendo de forma gradual, con oscilaciones apreciables, hasta asentarse en un rango medio.

4.7.3.4 Coste de entrenamiento y tamaño del modelo

Modelo	Épocas	Mejor época	Tiempo de entrenamiento (s)	Tiempo a mejor época (s)	Mejor Val MSE	Parámetros
ESN	24	9	23.00s	9.01s	0.00559	2001
KAN	100	85	38.45s	31.92s	0.000307	28
LSTM	19	4	10.08s	2.08s	0.000404	71297
LNN	147	132	438.67s	394.48s	0.000588	17921

Tabla 43. Coste de entrenamiento y tamaño del modelo para Google - Horizonte máximo.

Según los datos mostrados en la tabla 43, La KAN necesita unos 38 s totales y 28 parámetros para alcanzar su mejor validación; el ESN emplea algo menos de 23 s con 2 001 parámetros, pero su ajuste final es pobre. La LSTM tarda unos 10 s pese a contar con más de 71 000 pesos; la LNN es la más costosa: más de siete minutos de cómputo y cerca de 18 000 parámetros para lograr un error intermedio

4.8 Broadcom Inc. (AVGO)

4.8.1 Horizonte de 5 años

4.8.1.1 Resumen de métricas

Modelo	MSE	RMSE	MAE	R^2	MAPE (%)	SMAPE (%)
ESN	2409.14	49.08	43.51	-1.25	23.75%	27.49%
KAN	94.31	9.71	6.50	0.91	3.68%	3.69%
LSTM	1155.43	33.99	28.49	-0.08	15.24%	16.86%
LNN	3201.11	56.58	51.38	-2.00	28.27%	33.43%

Tabla 44. Resumen de métricas de modelos para Broadcom Inc. - Horizonte de 5 años.

Como se muestra en la tabla 44, KAN lidera con holgura (MAPE 3,68 %, R^2 0,91). LSTM queda lejos (15 %, $R^2 \approx 0$) y el ESN todavía peor (24 %, R^2 negativo). La LNN es la menos fiable, con un error cercano al 28 % y un R^2 de -2,00.

4.8.1.2 Precio real vs predicción

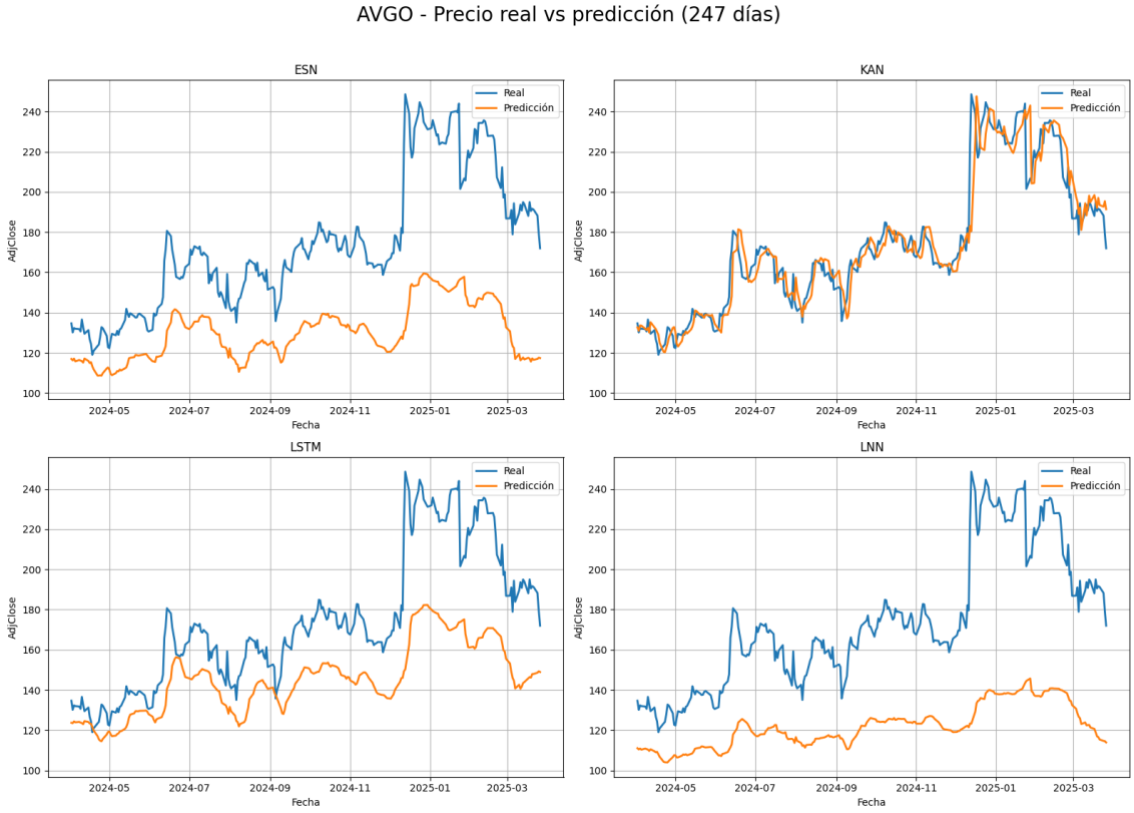

Figura 52. Precio real vs. predicción de los modelos para Broadcom Inc. - Horizonte de 5 años.

Como se observa en la figura 52, KAN acompaña de cerca cada tramo del precio, incluso el salto brusco a principios de 2025. LSTM sigue la forma general, pero se queda corto en los máximos. ESN infravalora casi todo el recorrido, y la LNN apenas refleja la tendencia, con una línea muy aplanada.

4.8.1.3 Curvas de pérdida

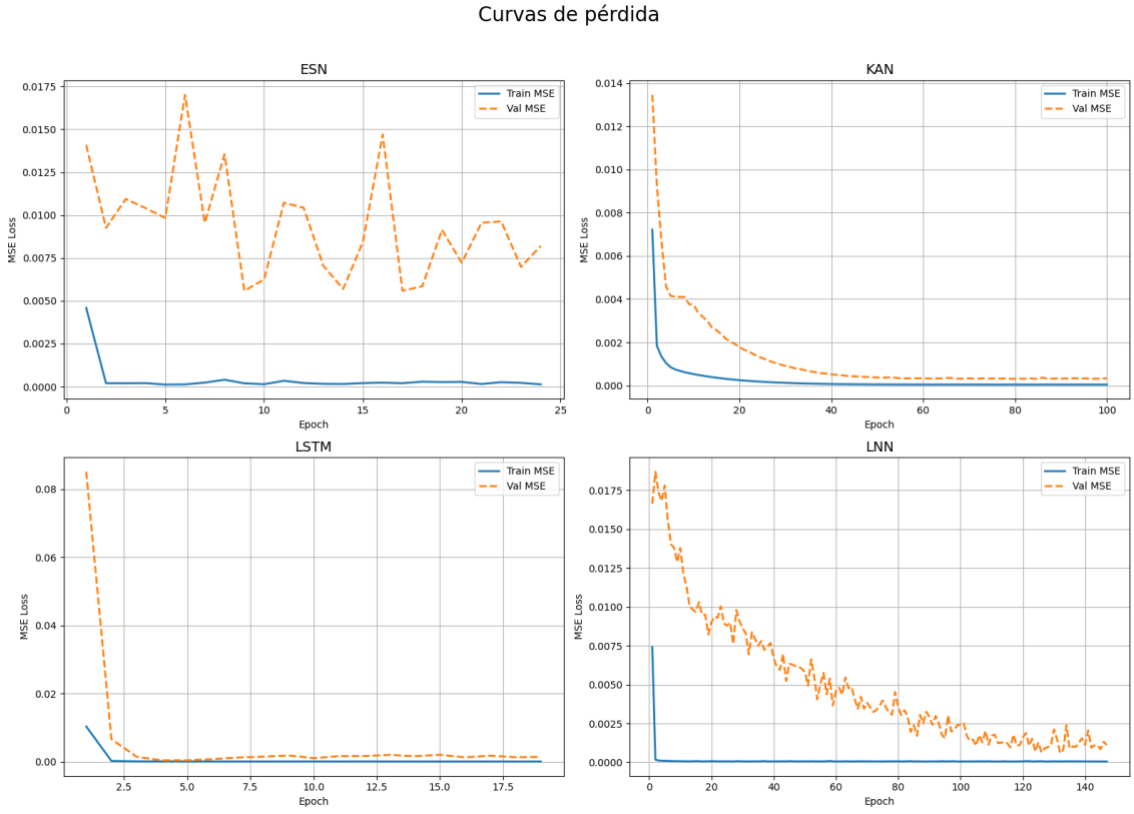

Figura 53. Curvas de pérdida de entrenamiento y validación de los modelos para Broadcom Inc. - Horizonte de 5 años.

Según los gráficos de la figura 53, KAN desciende rápido hasta una MSE de validación baja y estable. LSTM llega a un error igualmente bajo en pocos ciclos. En cambio, el ESN mantiene una validación alta y con picos frecuentes; la LNN parte de un error mucho mayor y lo reduce lentamente con oscilaciones continuas.

4.8.1.4 Coste de entrenamiento y tamaño del modelo

Modelo	Épocas	Mejor época	Tiempo de entrenamiento (s)	Tiempo a mejor época (s)	Mejor Val MSE	Parámetros
ESN	30	15	11.33s	5.77s	0.001517	2001
KAN	72	57	10.66s	8.44s	0.001147	28
LSTM	30	15	5.80s	3.08s	0.001686	71297
LNN	52	37	58.10s	41.21s	0.006731	17921

Tabla 45. Coste de entrenamiento y tamaño del modelo para Broadcom Inc. - Horizonte de 5 años.

Según los datos de la tabla 45, KAN ajusta sus 28 parámetros en unos 11 s; ESN tarda similar (11 s) pero con 2 001 parámetros y peor resultado. LSTM necesita apenas 6 s pese a sus 71 k pesos. LNN consume más de 58 s y 17 921 parámetros para un error muy superior.

4.8.2 Horizonte de 10 años

4.8.2.1 Resumen de métricas

Modelo	MSE	RMSE	MAE	R^2	MAPE (%)	SMAPE (%)
ESN	3800.72	61.65	46.71	-0.42	30.53%	38.69%
KAN	77.13	8.78	6.23	0.97	4.88%	5.05%
LSTM	1717.51	41.44	30.69	0.36	19.84%	23.06%
LNN	4643.19	68.14	53.43	-0.74	36.07%	46.79%

Tabla 46. Resumen de métricas de modelos para Broadcom Inc. - Horizonte de 10 años.

Según los datos de la tabla 46, KAN es el único modelo que mantiene un error contenido (MAPE 4,88 %; R^2 0,97). LSTM se aleja bastante (MAPE ≈ 20 %; R^2 0,36). El ESN y la LNN quedan muy lejos, con MAPE por encima del 30 % y R^2 negativos.

4.8.2.2 Precio real vs predicción

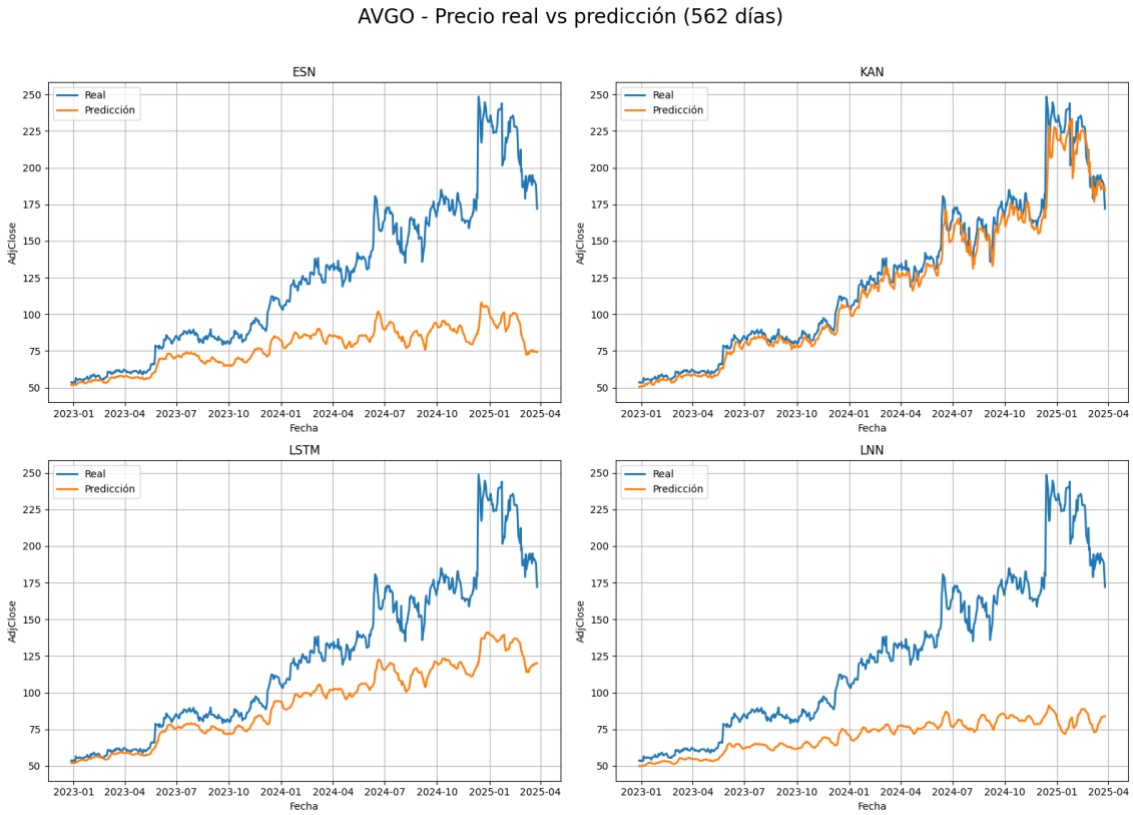

Figura 54. Precio real vs. predicción de los modelos para Broadcom Inc. - Horizonte de 10 años.

Según los gráficos de la figura 54, KAN acompaña la trayectoria del precio durante casi todo el periodo, incluso cuando la acción se dispara a principios de 2025. LSTM sigue la tendencia general, pero se queda claramente por debajo de la cotización real. ESN y LNN reproducen la forma global, aunque con valores muy deprimidos a lo largo de todo el tramo histórico.

4.8.2.3 Curvas de pérdida

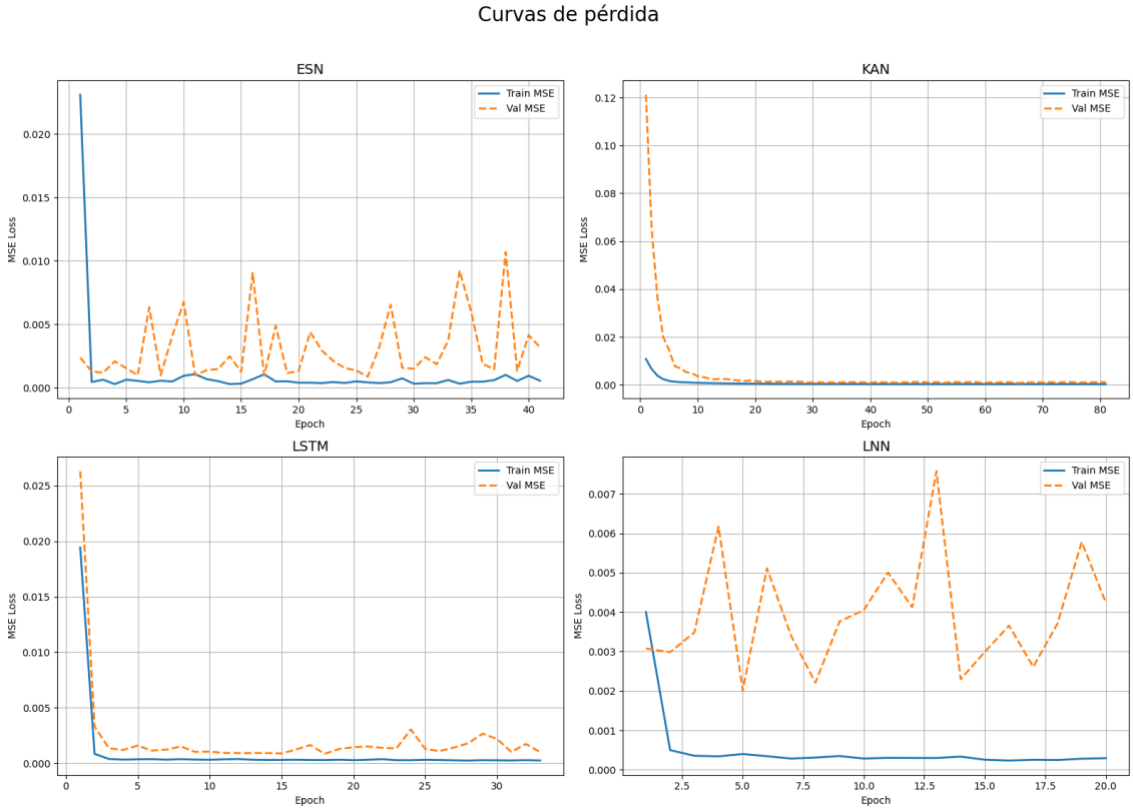

Figura 55. Curvas de pérdida de entrenamiento y validación de los modelos para Broadcom Inc. - Horizonte de 10 años.

Como se observa en la figura 55, la KAN desciende en pocas iteraciones hasta una MSE de validación mínima y estable. LSTM converg- e rápido, pero mantiene un nivel de error más alto que KAN. El ESN presenta validaciones con picos continuos durante todo el entrenamiento. La LNN parte de un error elevado y, aunque baja en los primeros ciclos, mantiene fuertes oscilaciones de validación después.

4.8.2.4 Coste de entrenamiento y tamaño del modelo

Modelo	Épocas	Mejor época	Tiempo de entrenamiento (s)	Tiempo a mejor época (s)	Mejor Val MSE	Parámetros
ESN	41	26	29.92s	19.14s	0.000882	2001
KAN	81	66	23.77s	19.42s	0.000781	28
LSTM	33	18	13.56s	7.24s	0.000851	71297
LNN	20	5	47.48s	12.07s	0.002012	17921

Tabla 47. Coste de entrenamiento y tamaño del modelo para Broadcom Inc. - Horizonte de 10 años.

Tal y como se muestra en la tabla 47, KAN logra su mejor validación en unos 19 s y solo 28 parámetros. El ESN tarda cerca de 20 s y maneja 2 001 parámetros, sin lograr buena precisión. LSTM necesita 13,5 s y gestiona más de 71 000 pesos; la LNN, aunque tiene menos parámetros, consume casi 48 s y no mejora el error de forma significativa.

4.8.3 Horizonte máximo

4.8.3.1 Resumen de métricas

Modelo	MSE	RMSE	MAE	R^2	MAPE (%)	SMAPE (%)
ESN	1434.92	37.88	23.71	0.53	15.39%	17.97%
KAN	335.21	18.31	14.80	0.89	13.50%	14.59%
LSTM	1298.63	36.04	21.98	0.57	13.96%	16.21%
LNN	3630.43	60.25	42.45	-0.19	30.31%	38.93%

Tabla 48. Resumen de métricas de modelos para Broadcom Inc. - Horizonte máximo.

Tal y como se muestra en la tabla 48, en la serie completa de Broadcom, ningún modelo logra un error bajo: la KAN es la "menos mala" (MAPE 13,5 %; R^2 0,89). LSTM queda cerca (13,96 %; R^2 0,57). El ESN sube al 15 % (R^2 0,53) y la LNN se desploma con un 30 % de error y R^2 negativo.

4.8.3.2 Precio real vs predicción

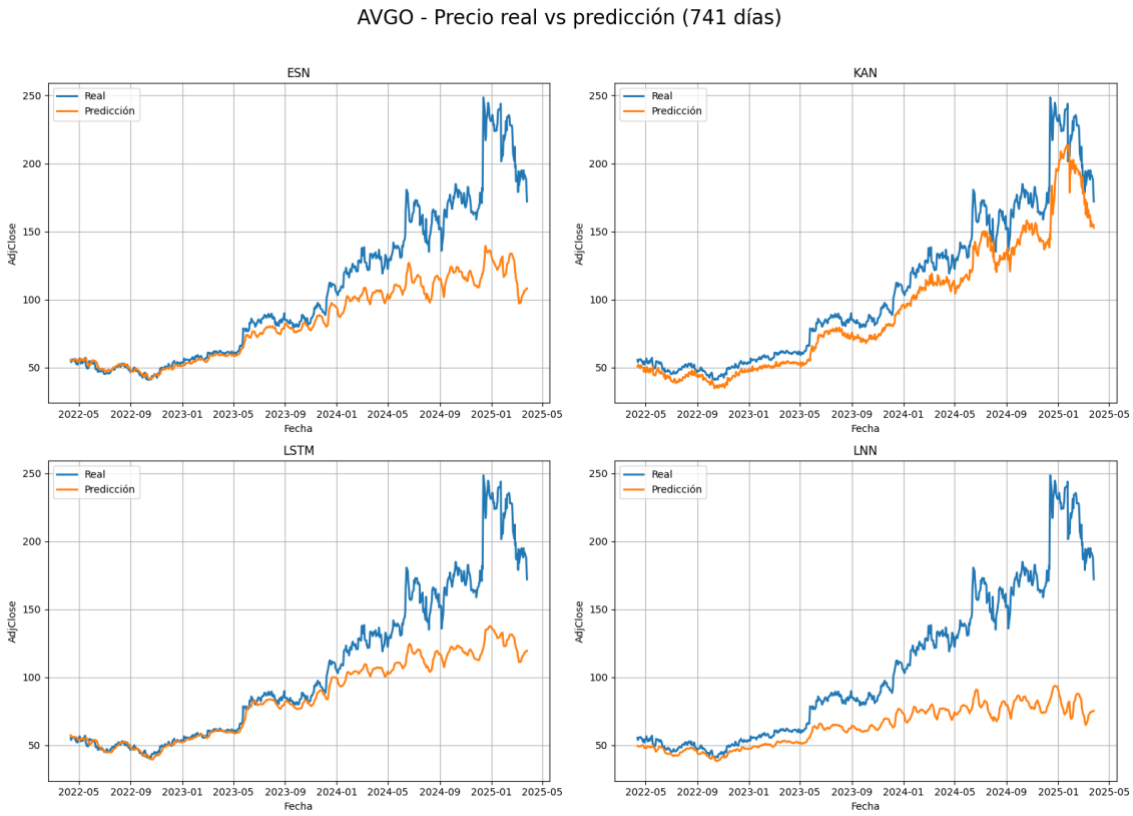

Figura 56. Precio real vs. predicción de los modelos para Broadcom Inc. - Horizonte máximo.

Tal y como se observa en la figura 56, la KAN sigue la pendiente general, aunque se queda algo corta en los extremos. LSTM traza una curva similar, ligeramente más baja. ESN reproduce la forma global pero siempre por debajo, y LNN apenas acompaña la tendencia: su línea naranja es mucho más plana que la azul original.

4.8.3.3 Curvas de pérdida

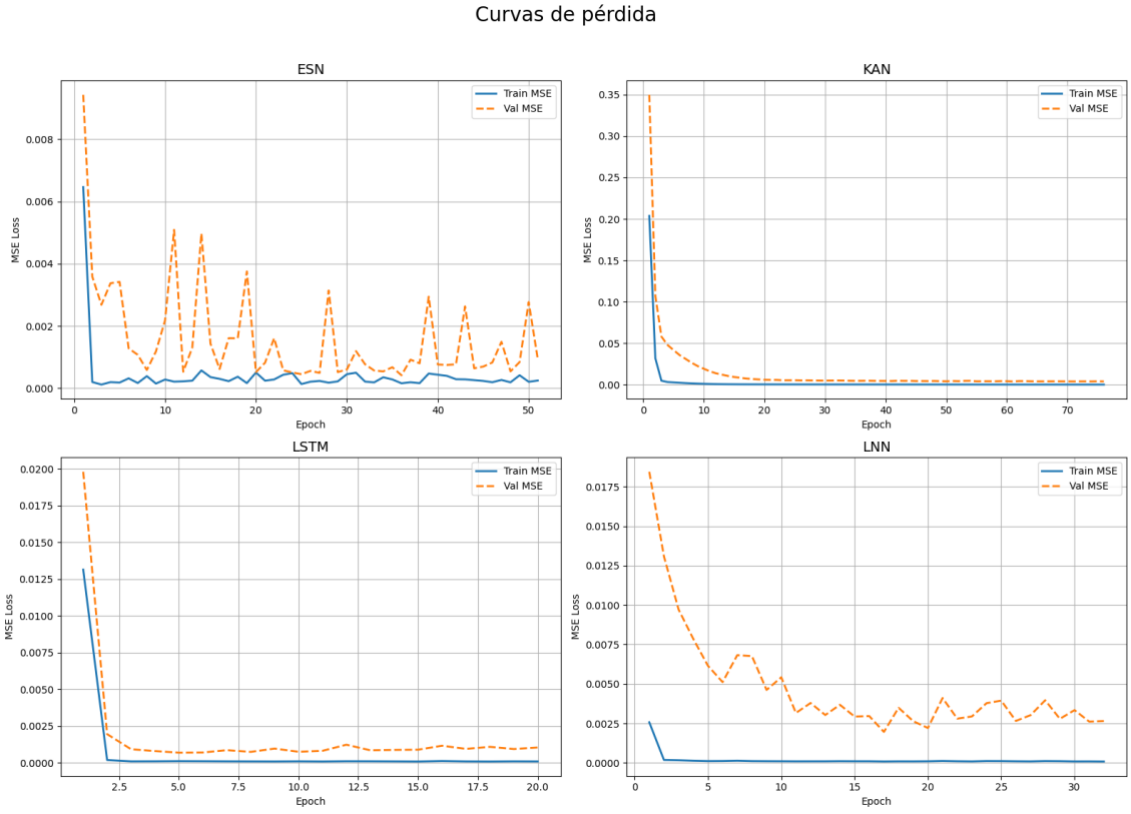

Figura 57. Curvas de pérdida de entrenamiento y validación de los modelos para Broadcom Inc. - Horizonte máximo.

Según los gráficos de la figura 57, la KAN baja rápido su MSE y se estabiliza, aunque parte de valores mucho más altos que en horizontes anteriores. LSTM converge con un perfil limpio; el ESN mantiene validaciones con picos frecuentes; la LNN desciende algo, pero conserva oscilaciones claras y nunca alcanza un error tan bajo como los demás.

4.8.3.4 Coste de entrenamiento y tamaño del modelo

Modelo	Épocas	Mejor época	Tiempo de entrenamiento (s)	Tiempo a mejor época (s)	Mejor Val MSE	Parámetros
ESN	51	36	48.58s	34.42s	0.00042	2001
KAN	76	61	30.73s	25.06s	0.003944	28
LSTM	20	5	10.45s	2.64s	0.000694	71297
LNN	32	17	98.43s	52.48s	0.001965	17921

Tabla 49. Coste de entrenamiento y tamaño del modelo para Broadcom Inc. - Horizonte máximo.

Tal y como aparece en la tabla 49 el ESN completa 51 ciclos en unos 49 s y ajusta 2 001 parámetros. La KAN tarda unos 31 s para 28 parámetros (mejor punto hacia el segundo 25). LSTM usa 10 s con 71 297 parámetros; la LNN, aun con menos pesos (17 921), necesita casi 100 s y no mejora el error final.

4.9 Berkshire Hathaway Inc. (BRK)

4.9.1 Horizonte de 5 años

4.9.1.1 Resumen de métricas

Modelo	MSE	RMSE	MAE	R^2	MAPE (%)	SMAPE (%)
ESN	620.52	24.91	21.99	0.39	4.76%	4.90%
KAN	52.79	7.27	5.43	0.95	1.20%	1.20%
LSTM	589.04	24.27	21.18	0.42	4.59%	4.72%
LNN	3165.06	56.26	52.43	-2.09	11.43%	12.20%

Tabla 50. Resumen de métricas de modelos para Berkshire Hathaway Inc. - Horizonte de 5 años.

Tal y como aparece en la tabla 50, KAN domina con claridad (MAPE 1,20 %, R^2 0,95). ESN y LSTM se mueven en torno al 4,7-4,6 % de error y $R^2 \approx 0,4$, mientras que la LNN queda muy atrás (MAPE 11 %, R^2 negativo).

4.9.1.2 Precio real vs predicción

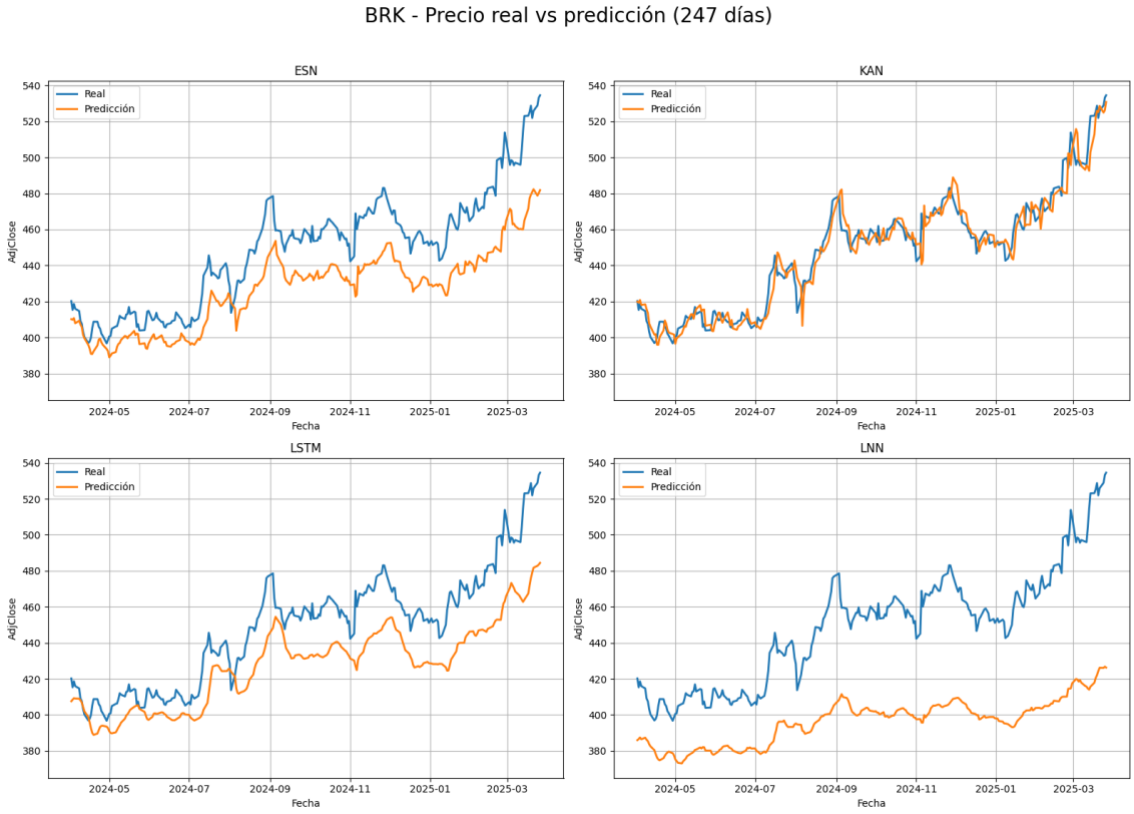

Figura 58. Precio real vs. predicción de los modelos para Berkshire Hathaway Inc. - Horizonte de 5 años.

Según los gráficos de la figura 58, KAN reproduce casi punto por punto la cotización, incluso el empuje alcista de febrero-marzo 2025. ESN y LSTM siguen la tendencia, pero subestiman los máximos; LNN queda claramente por debajo durante todo el periodo.

4.9.1.3 Curvas de pérdida

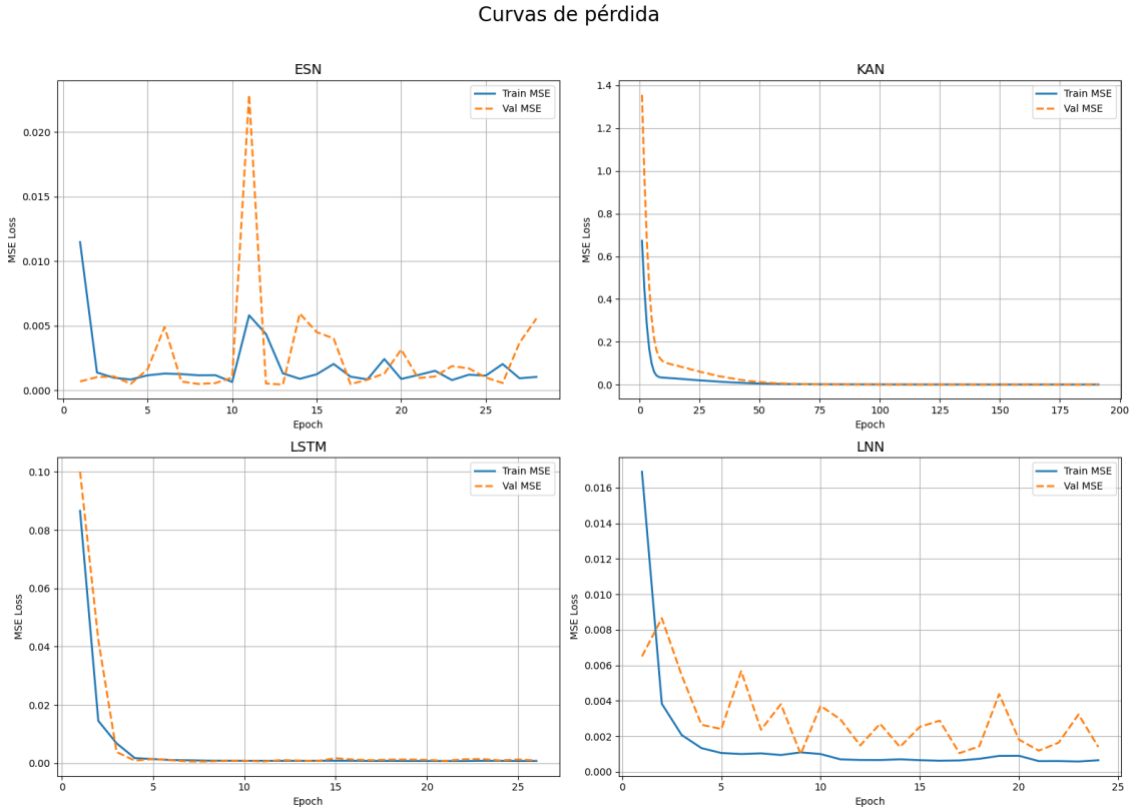

Figura 59. Curvas de pérdida de entrenamiento y validación de los modelos para Berkshire Hathaway Inc. - Horizonte de 5 años.

Según los gráficos de la figura 59, KAN desciende rápido a una MSE mínima y estable. LSTM presenta un perfil similar, sin repuntes. ESN muestra varios picos de validación, con uno especialmente alto en la mitad del entrenamiento. LNN baja el error al principio y luego oscila de forma apreciable.

4.9.1.4 Coste de entrenamiento y tamaño del modelo

Modelo	Épocas	Mejor época	Tiempo de entrenamiento (s)	Tiempo a mejor época (s)	Mejor Val MSE	Parámetros
ESN	28	13	10.54s	4.95s	0.000439	2001
KAN	191	176	27.23s	25.19s	0.000413	28
LSTM	26	11	5.04s	2.18s	0.000574	71297
LNN	24	9	28.70s	10.93s	0.001001	17921

Tabla 51. Coste de entrenamiento y tamaño del modelo para Berkshire Hathaway Inc. - Horizonte de 5 años.

Tal y como se muestra en la tabla 51, ESN completa su ajuste en unos 11 s con 2 001 parámetros. KAN tarda unos 27 s (28 parámetros) y alcanza la mejor validación casi al final. LSTM necesita apenas 5 s pero maneja 71 k pesos. LNN es el más lento: ~29 s y 17 921 parámetros para un error mucho mayor.

4.9.2 Horizonte de 10 años

4.9.2.1 Resumen de métricas

Modelo	MSE	RMSE	MAE	R^2	MAPE (%)	SMAPE (%)
ESN	480.95	21.93	16.32	0.86	3.75%	3.87%
KAN	35.82	5.99	4.54	0.99	1.14%	1.14%
LSTM	59.80	7.73	5.69	0.98	1.39%	1.40%
LNN	520.32	22.81	17.18	0.85	3.97%	4.09%

Tabla 52. Resumen de métricas de modelos para Berkshire Hathaway Inc. - Horizonte de 10 años.

Según la tabla 52, La KAN vuelve a liderar (MAPE 1,14 %; R^2 0,99). La LSTM le sigue de cerca (1,39 %; 0,98). ESN y LNN se sitúan en torno al 3,8-4 % de error con $R^2 \approx 0,86$.

4.9.2.2 Precio real vs predicción

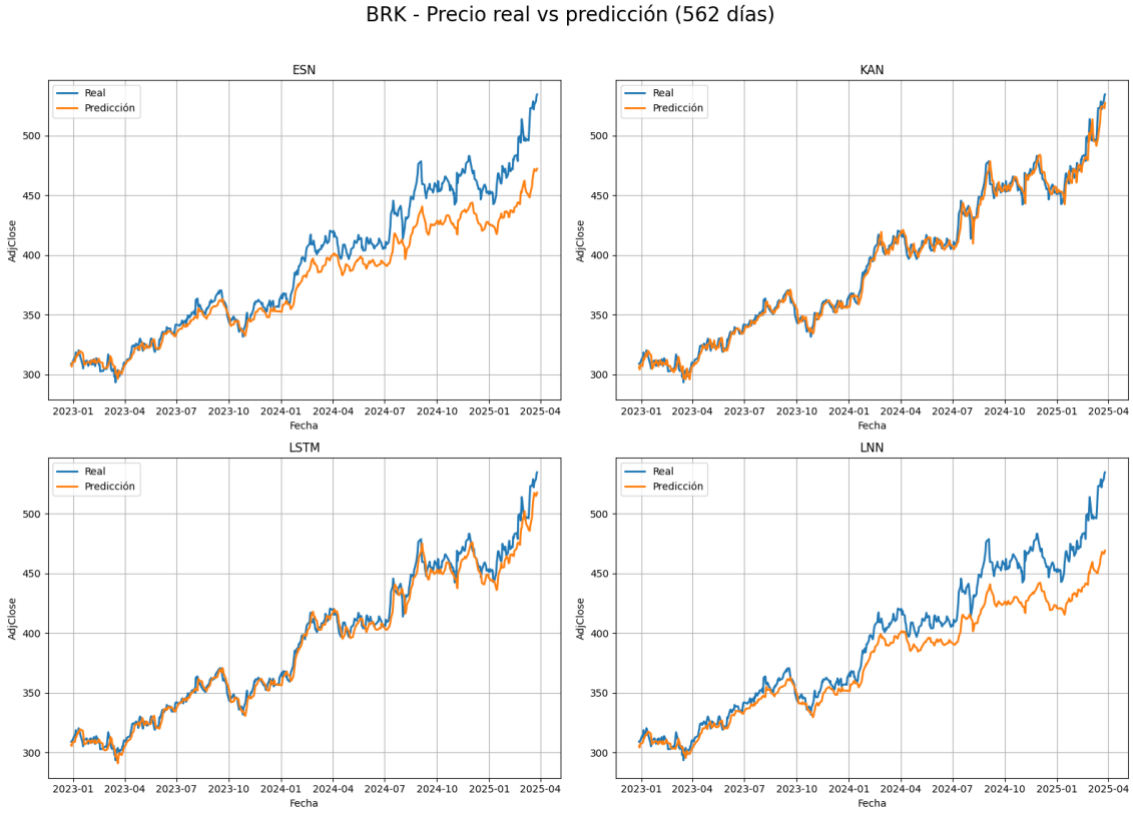

Figura 60. Precio real vs. predicción de los modelos para Berkshire Hathaway Inc. - Horizonte de 10 años.

Como se observa en la figura 60, KAN reproduce casi al milímetro toda la trayectoria, incluidos los escalones de marzo 2024 y febrero 2025. LSTM acompaña la tendencia con leves suavizados de picos. ESN subestima sistemáticamente los máximos, mientras que LNN se queda algo corta en la zona alta del rango.

4.9.2.3 Curvas de pérdida

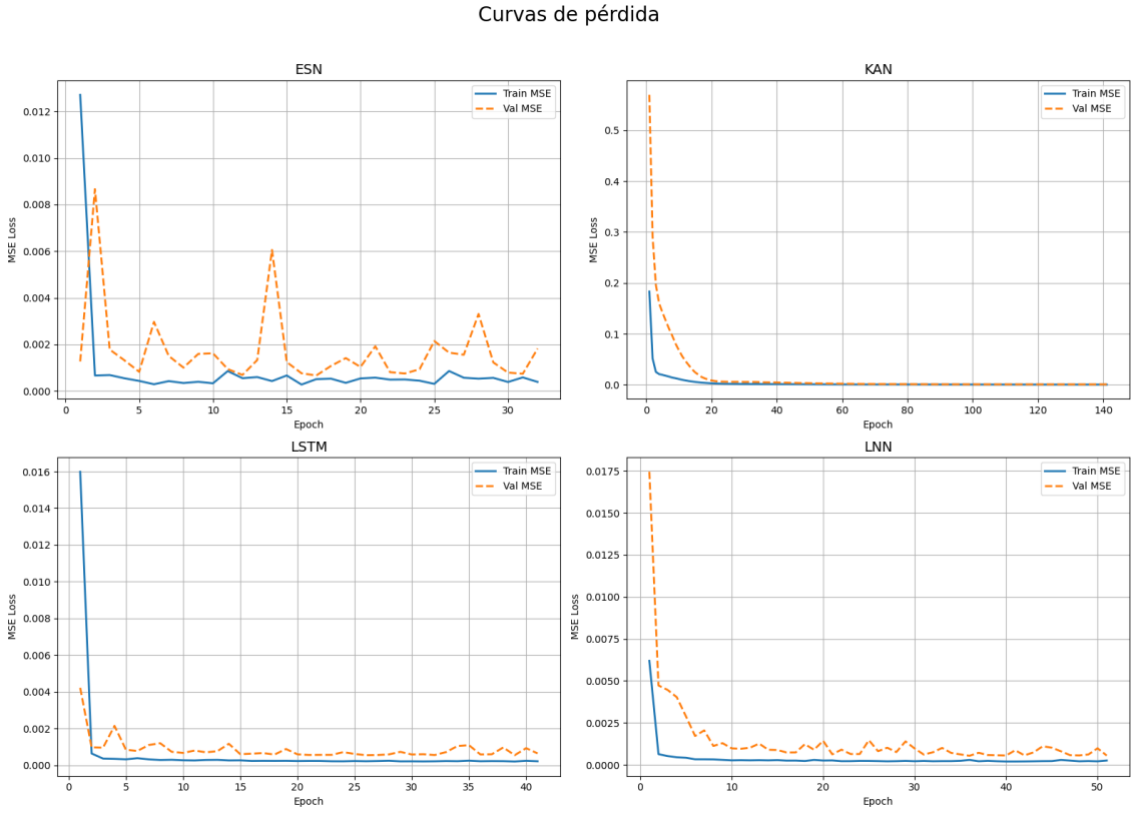

Figura 61. Curvas de pérdida de entrenamiento y validación de los modelos para Berkshire Hathaway Inc. - Horizonte de 10 años.

Como se observa en la figura 61, KAN desciende rápido a una MSE de validación mínima y estable; LSTM muestra un perfil parecido sin sobresaltos. ESN mantiene repuntes de validación a lo largo del entrenamiento. LNN baja pronto el error y luego oscila en un rango estrecho.

4.9.2.4 Coste de entrenamiento y tamaño del modelo

Modelo	Épocas	Mejor época	Tiempo de entrenamiento (s)	Tiempo a mejor época (s)	Mejor Val MSE	Parámetros
ESN	32	17	23.63s	12.84s	0.00067	2001
KAN	141	126	40.98s	36.76s	0.000528	28
LSTM	41	26	16.70s	10.54s	0.000547	71297
LNN	51	36	121.02s	86.15s	0.00054	17921

Tabla 53. Coste de entrenamiento y tamaño del modelo para Berkshire Hathaway Inc. - Horizonte de 10 años.

Como muestra la tabla 53, ESN concluye en 24 s con 2 001 parámetros. KAN necesita unos 41 s pero solo 28 parámetros. LSTM ajusta sus 71 297 pesos en 17 s; la LNN es la más lenta, 121 s para 17 921 parámetros.

4.9.3 Horizonte máximo

4.9.3.1 Resumen de métricas

Modelo	MSE	RMSE	MAE	R^2	MAPE (%)	SMAPE (%)
ESN	962.88	31.03	23.34	0.78	5.63%	5.89%
KAN	33.83	5.82	4.42	0.99	1.22%	1.22%
LSTM	353.06	18.79	13.73	0.92	3.33%	3.41%
LNN	1172.86	34.25	26.07	0.74	6.30%	6.62%

Tabla 54. Resumen de métricas de modelos para Berkshire Hathaway Inc. - Horizonte máximo.

Según los datos de la tabla 54, KAN vuelve a ser la referencia con un MAPE de 1,22 % y R^2 = 0,99. LSTM mantiene buenos números (3,33 %; 0,92). ESN y LNN se quedan claramente atrás, con errores del 5-6 % y R^2 por debajo de 0,80.

4.9.3.2 Precio real vs predicción

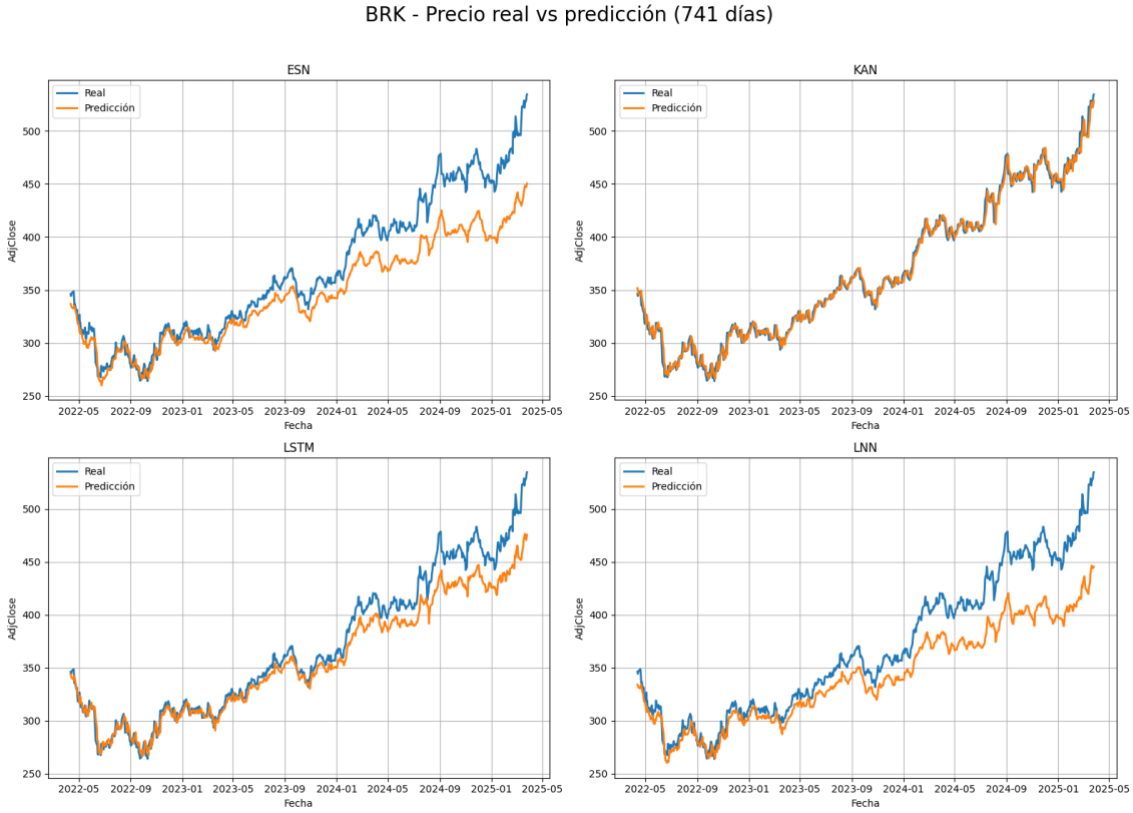

Figura 62. Precio real vs. predicción de los modelos para Berkshire Hathaway Inc. - Horizonte máximo.

Tal y como se observa en la figura 62, en los 741 días de test KAN sigue casi al milímetro la trayectoria de Berkshire, incluidas las fuertes subidas de finales de 2024 y marzo 2025. LSTM acompaña bien la pendiente, aunque se queda algo corta en los techos. ESN reproduce la forma general, pero siempre por debajo de los máximos, y la LNN resulta la más "plana", subestimando buena parte del rango alto.

4.9.3.3 Curvas de pérdida

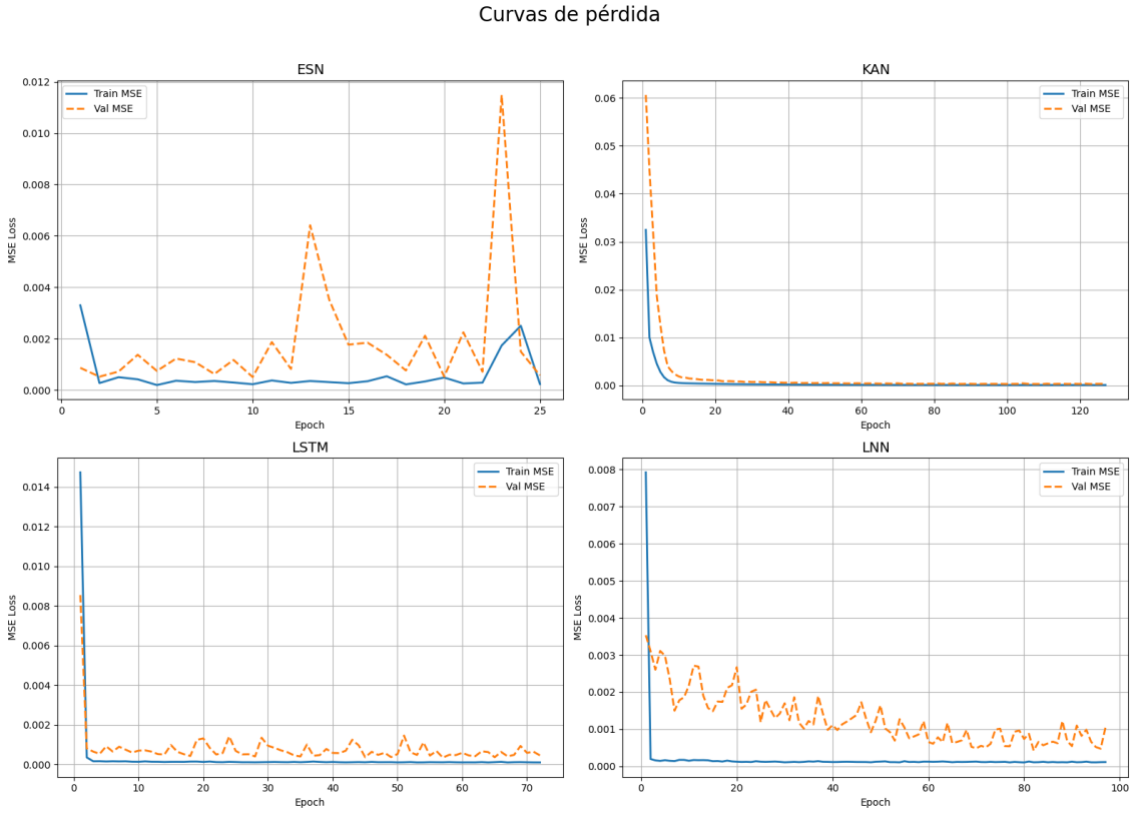

Figura 63. Curvas de pérdida de entrenamiento y validación de los modelos para Berkshire Hathaway Inc. - Horizonte máximo.

Como se observa en la figura 63, KAN baja la MSE de validación en pocas iteraciones y luego se mantiene estable. LSTM muestra un perfil similar sin picos relevantes. ESN presenta repuntes frecuentes mientras que la LNN oscila durante todo el entrenamiento, sin lograr asentarse tan bajo como KAN o LSTM.

4.9.3.4 Coste de entrenamiento y tamaño del modelo

Modelo	Épocas	Mejor época	Tiempo de entrenamiento (s)	Tiempo a mejor época (s)	Mejor Val MSE	Parámetros
ESN	25	10	23.81s	9.66s	0.000503	2001
KAN	127	112	46.99s	41.48s	0.00032	28
LSTM	72	57	37.40s	29.63s	0.000349	71297
LNN	97	82	291.83s	246.96s	0.000425	17921

Tabla 55. Coste de entrenamiento y tamaño del modelo para Berkshire Hathaway Inc. - Horizonte máximo.

Tal y como aparece en la tabla 55, KAN necesita unos 47 s en total y solo 28 parámetros (mejor punto alrededor de los 41 s). ESN concluye en 24 s con 2 001 parámetros, pero con peor precisión final. LSTM emplea 37 s para ajustar 71 297 pesos y obtiene un buen resultado. La LNN es la más lenta: casi 5 min de cómputo y 17 921 parámetros para un error que no mejora al ESN.

4.10 Eli Lilly and Company (LLY)

4.10.1 Horizonte de 5 años

4.10.1.1 Resumen de métricas

Modelo	MSE	RMSE	MAE	R^2	MAPE (%)	SMAPE (%)
ESN	7662.45	87.54	80.22	-0.80	9.36%	9.89%
KAN	657.99	25.65	18.88	0.85	2.30%	2.28%
LSTM	3442.40	58.67	42.62	0.19	5.07%	5.29%
LNN	45643.39	213.64	208.87	-9.74	24.83%	28.50%

Tabla 56. Resumen de métricas de modelos para Eli Lilly and Company - Horizonte de 5 años.

Tal y como aparece en la tabla 56, KAN vuelve a ser el modelo más fiable (MAPE ≈ 2,3 %, R^2 0,85). LSTM se mueve en torno al 5 % de error y mantiene R^2 positivo (0,19). El ESN sube al 9 % con R^2 negativo (-0,80) y la LNN se dispara a un 25 % de error con un R^2 muy por debajo de cero.

4.10.1.2 Precio real vs predicción

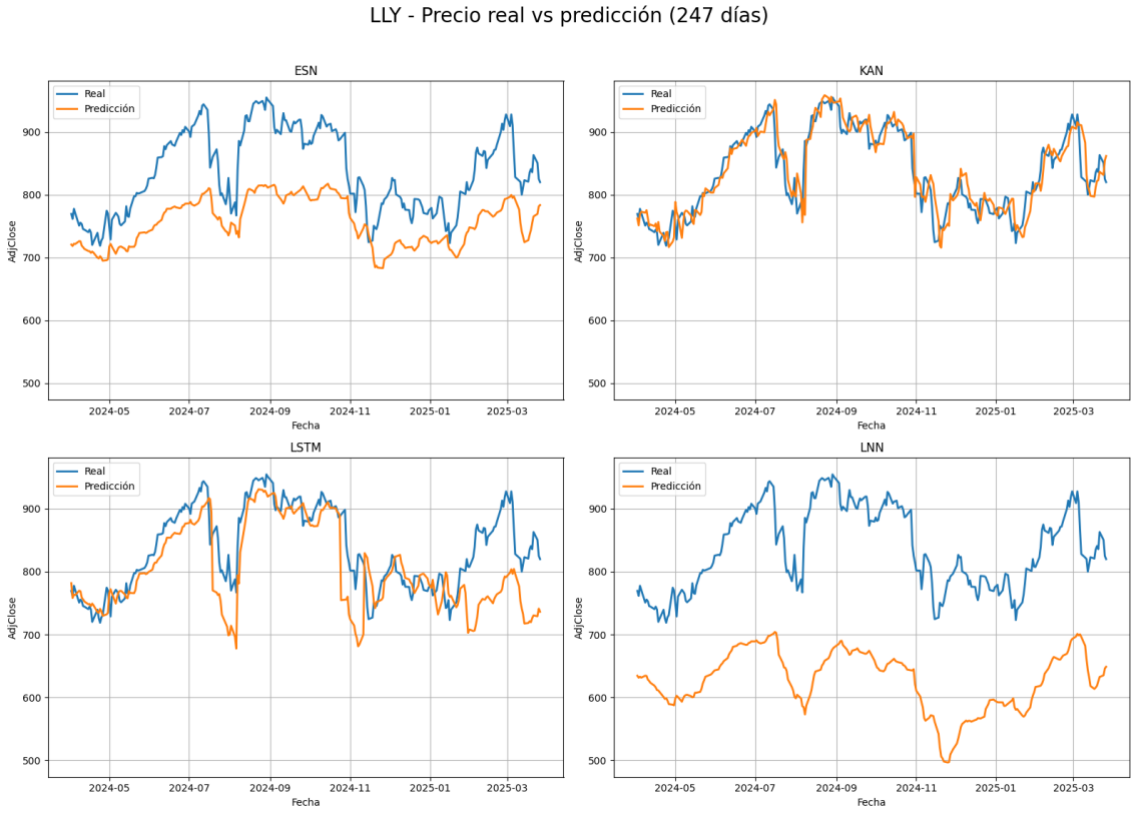

Figura 64. Precio real vs. predicción de los modelos para Eli Lilly and Company - Horizonte de 5 años.

Según los gráficos de la figura 64, KAN sigue de forma bastante fiel las oscilaciones de Eli Lilly, incluidas las crestas de agosto y enero. LSTM acompaña la tendencia, pero se queda corto en los picos y en las caídas profundas. ESN mantiene la forma general, aunque reproduce valores sensiblemente más bajos a lo largo de todo el tramo. LNN es el que más se desmarca, con una curva muy aplanada frente a la volatilidad real.

4.10.1.3 Curvas de pérdida

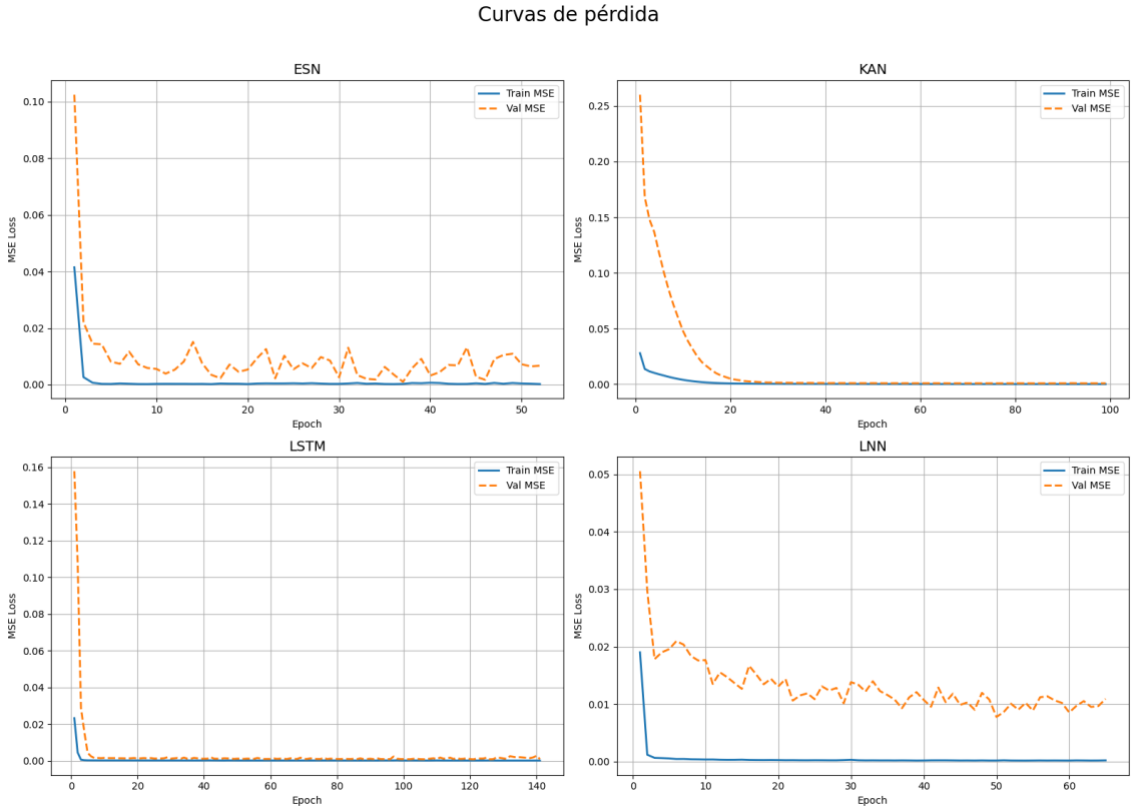

Figura 65. Precio real vs. predicción de los modelos para Eli Lilly and Company - Horizonte de 5 años.

Según los gráficos de la figura 65, KAN reduce su MSE de validación en las primeras iteraciones y se estabiliza. LSTM muestra un patrón igualmente estable, sin repuntes significativos. ESN alcanza un nivel bajo de error de entrenamiento, pero su validación fluctúa durante todo el ajuste. LNN desciende al principio y luego mantiene oscilaciones amplias, reflejando un ajuste inestable.

4.10.1.4 Coste de entrenamiento y tamaño del modelo

Modelo	Épocas	Mejor época	Tiempo de entrenamiento (s)	Tiempo a mejor época (s)	Mejor Val MSE	Parámetros
ESN	52	37	18.72s	13.42s	0.001	2001
KAN	99	84	14.09s	11.85s	0.000882	28
LSTM	141	126	27.63s	24.66s	0.000844	71297
LNN	65	50	74.54s	57.79s	0.007756	17921

Tabla 57. Coste de entrenamiento y tamaño del modelo para Eli Lilly and Company - Horizonte de 5 años.

Según los datos de la tabla 57, ESN completa 52 épocas en unos 19 s con 2 001 parámetros. KAN necesita 14 s para 28 parámetros y obtiene su mejor validación en 12 s. LSTM tarda 28 s para entrenar 71 297 parámetros, y la LNN consume 75 s con 17 921 parámetros, sin traducir ese coste en precisión.

4.10.2 Horizonte de 10 años

4.10.2.1 Resumen de métricas

Modelo	MSE	RMSE	MAE	R^2	MAPE (%)	SMAPE (%)
ESN	43441.00	208.43	171.14	-0.11	22.55%	26.52%
KAN	423.45	20.58	13.82	0.99	2.07%	2.06%
LSTM	19214.60	138.62	109.73	0.51	14.18%	15.77%
LNN	120148.13	346.62	298.14	-2.08	40.78%	53.89%

Tabla 58. Resumen de métricas de modelos para Eli Lilly and Company - Horizonte de 10 años.

Según los datos de la tabla 58, en esta ventana ampliada KAN conserva un ajuste prácticamente perfecto (MAPE 2,07 %, R^2 0,99). La LSTM mejora mucho respecto al horizonte corto, pero sigue lejos (14 %, R^2 0,51). El ESN sube al 22 % de error y muestra R^2 levemente negativo, mientras que la LNN naufraga con un 41 % de MAPE y R^2 -2,08.

4.10.2.2 Precio real vs predicción

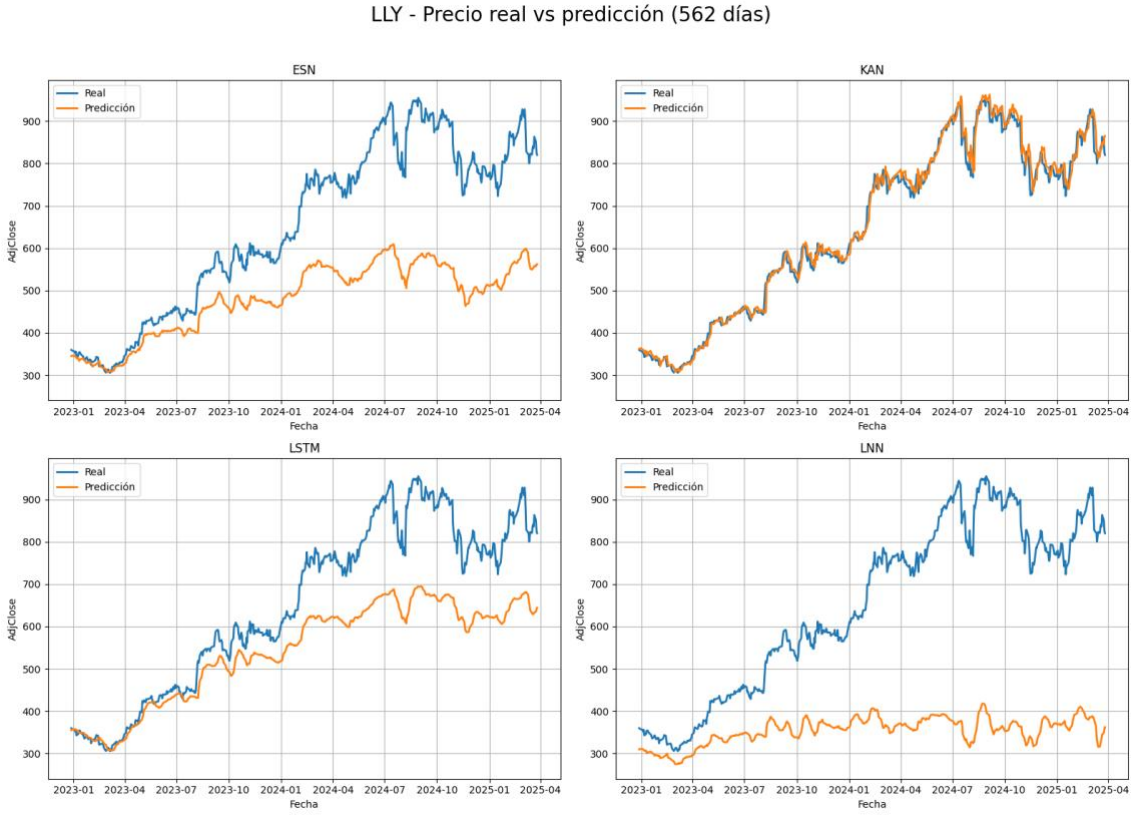

Figura 66. Precio real vs. predicción de los modelos para Eli Lilly and Company - Horizonte de 10 años.

Tal y como se observa en la figura 66, la curva de KAN se superpone casi por completo a la serie real, incluso en las fuertes subidas de verano de 2024 y los retrocesos de marzo 2025. La LSTM acompaña la pendiente general, aunque se queda corta en los techos y suelos más marcados. El ESN sigue la tendencia, pero con valores sistemáticamente por debajo; la LNN apenas refleja la dinámica, con una línea muy aplanada frente a la volatilidad real.

4.10.2.3 Curvas de pérdida

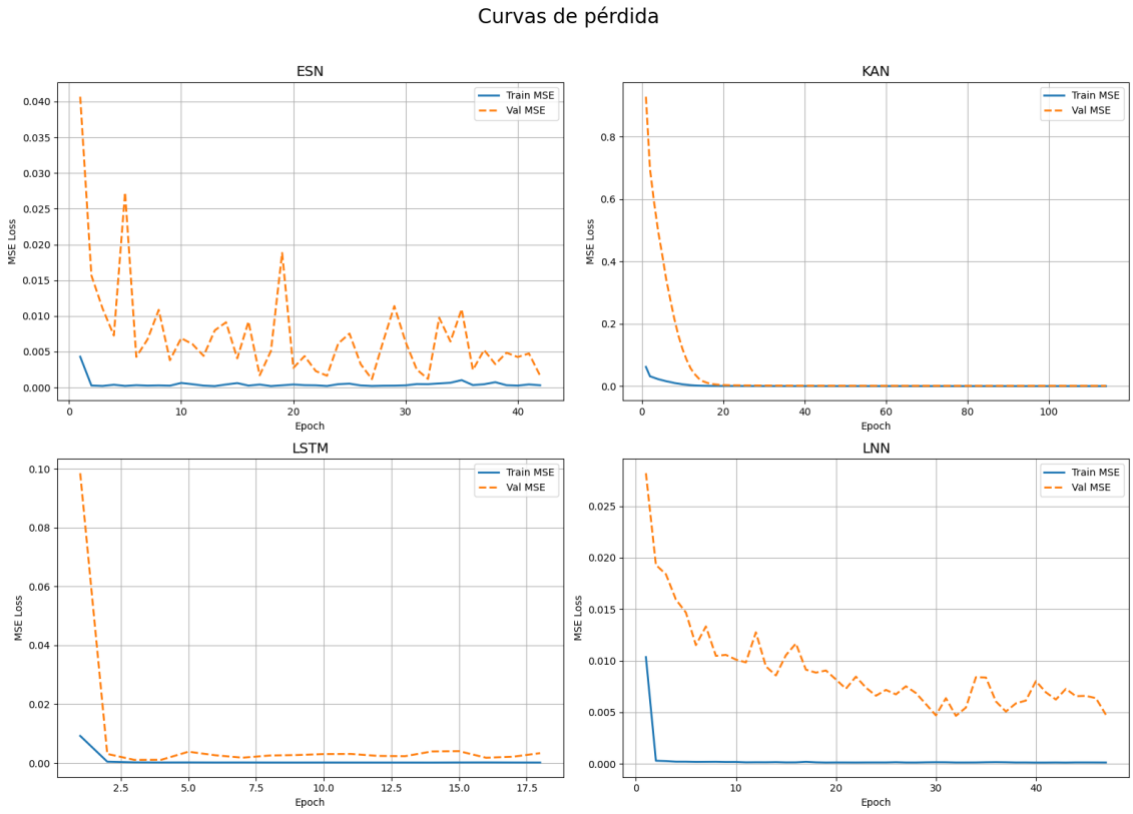

Figura 67. Curvas de pérdida de entrenamiento y validación de los modelos Eli Lilly and Company - Horizonte de 10 años.

Tal y como se observa en la figura 67, KAN reduce el error de validación con rapidez y se mantiene plano el resto del entrenamiento. La LSTM muestra un patrón muy estable tras la caída inicial. El ESN mantiene un nivel de validación alto y plagado de picos; la LNN desciende en los primeros ciclos y luego oscila en un rango mucho mayor que los otros modelos.

4.10.2.4 Coste de entrenamiento y tamaño del modelo

Modelo	Épocas	Mejor época	Tiempo de entrenamiento (s)	Tiempo a mejor época (s)	Mejor Val MSE	Parámetros
ESN	42	27	30.79s	20.03s	0.001164	2001
KAN	114	99	33.31s	29.06s	0.000792	28
LSTM	18	3	7.27s	1.20s	0.001059	71297
LNN	47	32	112.99s	77.81s	0.004663	17921

Tabla 59. Coste de entrenamiento y tamaño del modelo para Eli Lilly and Company - Horizonte de 10 años.

Como se muestra en la tabla 59, ESN completó 42 épocas en unos 31 s, fijando su mejor punto cerca del segundo 20 con 2 001 pesos. KAN necesitó 114 épocas y 33 s en total para lograr la mínima MSE. La LSTM entrenó 18 épocas en poco más de 7 s pese a sus 71 297 pesos. LNN fue el más costoso: 47 épocas y más de 113 s de cómputo con 17 921 parámetros, sin traducir ese tiempo en precisión.

4.10.3 Horizonte máximo

4.10.3.1 Resumen de métricas

Modelo	MSE	RMSE	MAE	R^2	MAPE (%)	SMAPE (%)
ESN	75052.80	273.96	215.66	-0.49	31.24%	39.44%
KAN	460.89	21.47	14.23	0.99	2.41%	2.37%
LSTM	29892.23	172.89	130.82	0.41	18.35%	21.15%
LNN	116839.92	341.82	282.58	-1.32	43.15%	57.81%

Tabla 60. Resumen de métricas de modelos para Eli Lilly and Company - Horizonte máximo.

Como se muestra en la tabla 60, KAN vuelve a sobresalir con un MAPE de 2,41 % y $R^2 \approx 0{,}99$. LSTM queda lejos (18 %; $R^2 \approx 0{,}41$) y el ESN se dispara al 31 % de error con R^2 negativo. La LNN es la peor: 43 % de MAPE y $R^2 \approx -1{,}3$.

4.10.3.2 Precio real vs predicción

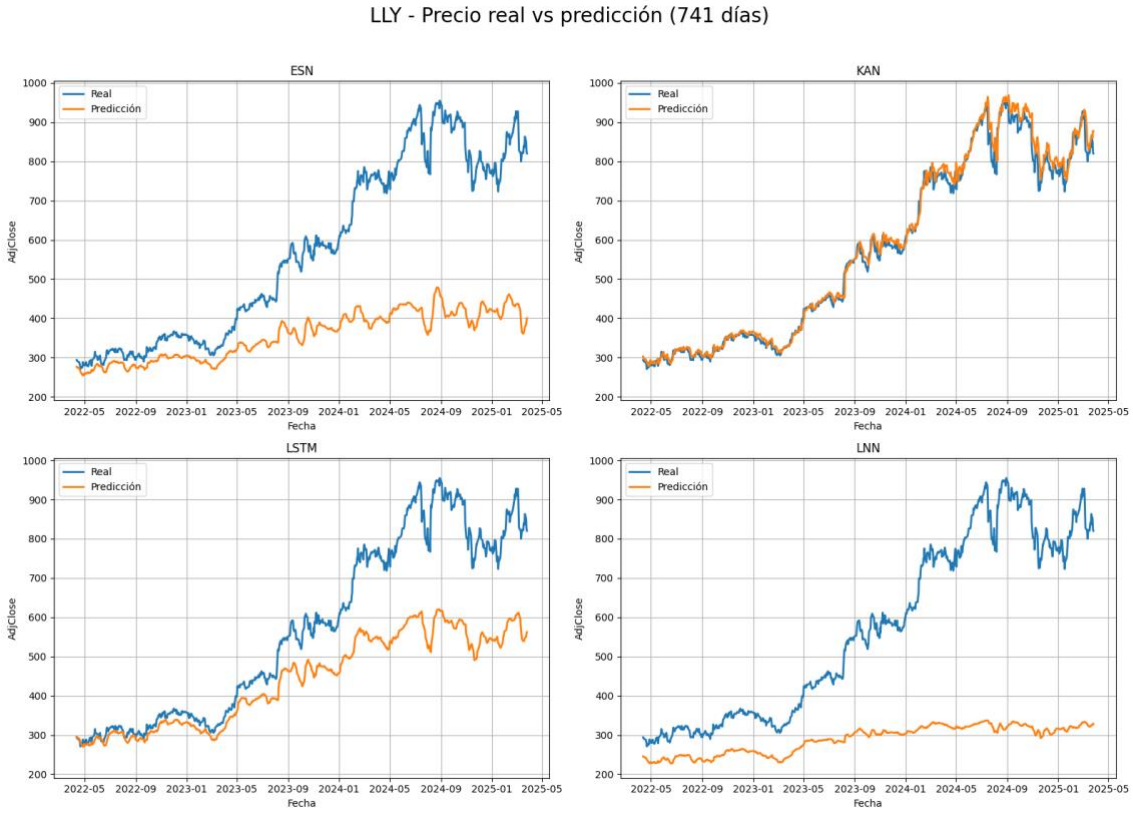

Figura 68. Precio real vs. predicción de los modelos para Eli Lilly and Company - Horizonte máximo.

Según los gráficos de la figura 68, KAN reproduce casi al detalle la evolución completa, incluso el tramo alcista de 2024 y la corrección de 2025. LSTM acompaña la pendiente general, pero recorta todos los picos. ESN subestima de forma clara toda la serie, y la LNN apenas recoge la forma: su curva es casi plana frente a la volatilidad real.

4.10.3.3 Curvas de pérdida

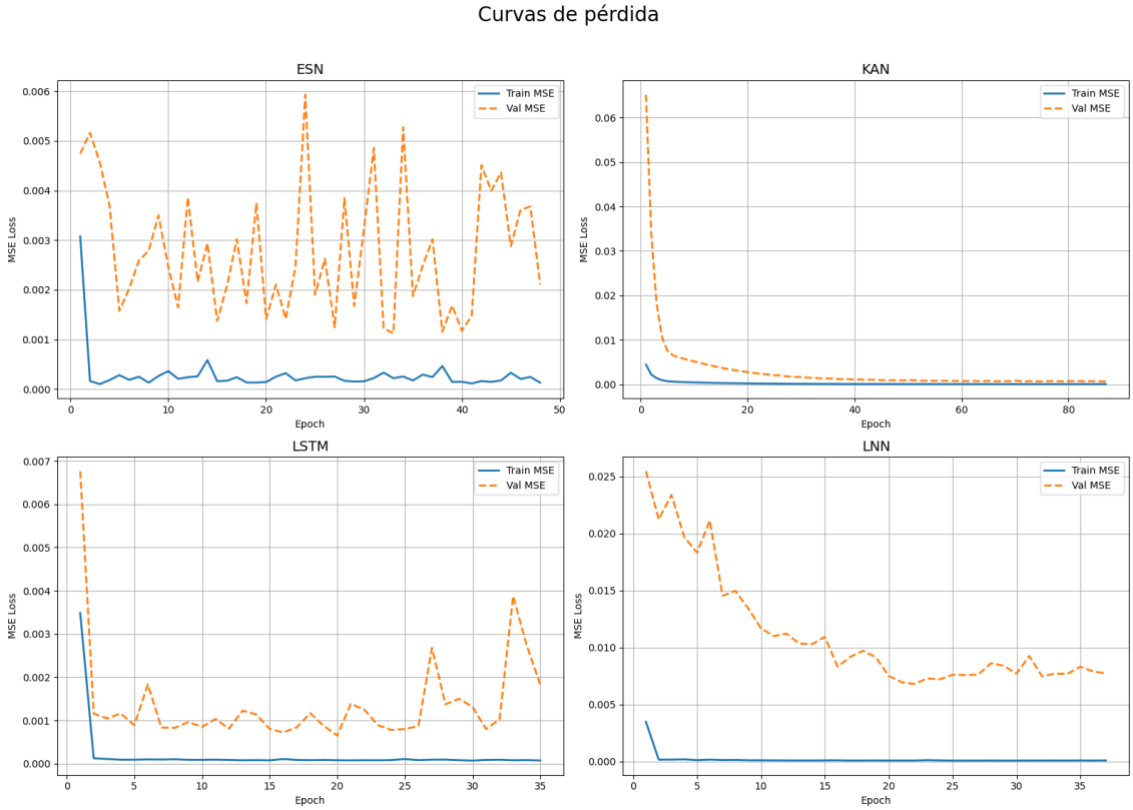

Figura 69. Curvas de pérdida de entrenamiento y validación de los modelos Eli Lilly and Company - Horizonte máximo.

Según los gráficos de la figura 69, KAN desciende enseguida a una MSE de validación mínima y se mantiene estable. LSTM baja rápido, pero muestra repuntes moderados hacia el final. ESN conserva validaciones altas con picos continuos; la LNN arranca en un error muy elevado y, aunque desciende, lo hace con oscilaciones marcadas.

4.10.3.4 Coste de entrenamiento y tamaño del modelo

Modelo	Épocas	Mejor época	Tiempo de entrenamiento (s)	Tiempo a mejor época (s)	Mejor Val MSE	Parámetros
ESN	48	33	44.44s	30.50s	0.001122	2001
KAN	87	72	32.30s	26.86s	0.00068	28
LSTM	35	20	18.36s	10.41s	0.000651	71297
LNN	37	22	112.57s	68.01s	0.006788	17921

Tabla 61. Coste de entrenamiento y tamaño del modelo para Eli Lilly and Company - Horizonte máximo.

Como se muestra en la tabla 61, KAN necesita unos 32 s (mejor punto al segundo 27) manejando solo 28 parámetros. ESN tarda 44 s con 2 001 pesos, pero su precisión es muy inferior. LSTM emplea 18 s para entrenar 71 297 parámetros, y la LNN es la más lenta: 113 s y 17 921 parámetros, sin traducir ese coste en un buen resultado.

4.11 Interpretación de resultados

A lo largo del capítulo se han evaluado cuatro arquitecturas (ESN, KAN, LSTM y LNN) sobre diez valores del S&P 500 y en tres horizontes (≈ 5, 10 y 13 años). El conjunto de números, gráficas y tiempos de cómputo revela patrones bastante coherentes que permiten extraer conclusiones generales y puntuales.

La Kolmogorov-Arnold Network (KAN) se sitúa siempre en cabeza. Ofrece el menor error porcentual y el mayor coeficiente R^2, incluso cuando las cotizaciones se mueven con gran brusquedad, como sucede en Nvidia, Broadcom o Tesla. Su tamaño diminuto (28 parámetros) hace que los tiempos de entrenamiento ronden entre medio minuto y un minuto y, además, evita casi por completo el sobreajuste ya que la curva de validación desciende rápido y luego permanece estable.

Detrás está la LSTM. En series relativamente suaves, como las de Google o Microsoft, llega a igualar a la KAN en la ventana de diez años. Sin embargo, cuando la trayectoria del precio incluye saltos repentinos, el modelo tiende a suavizar los extremos y su error crece tres o cuatro puntos.

La ESN mantiene un rendimiento aceptable mientras la variabilidad no sea demasiado alta, pero pierde precisión cuando la ventana se amplía o la serie cambia bruscamente; el caso de Eli Lilly ilustra bien esta debilidad, ya que su MAPE pasa de alrededor de un 9 % a más de un 30 % entre cinco y trece años. Su punto fuerte es el coste ya que podemos observar que con poco más de dos mil pesos suele terminar el ajuste en menos de veinte segundos.

La LNN es la alternativa más irregular. Solo en contadas ocasiones adelanta a LSTM o ESN; en la mayoría de los escenarios, y sobre todo con Broadcom y Eli Lilly, queda muy por detrás. Sus curvas de validación oscilan sin llegar a asentarse y el tiempo de cálculo puede multiplicar por cinco al de la KAN sin compensarlo en precisión.

Ampliar de cinco a diez años suele beneficiar a todos los modelos, porque añade contexto sin provocar grandes cambios estructurales. El salto a la serie completa, en cambio, separa claramente a las redes, ya que vemos que KAN y LSTM mantienen su rendimiento (su MAPE crece uno o dos puntos como mucho), mientras que ESN y LNN evidencian un rendimiento menor. En la LNN el error llega a rondar el 40 % en Broadcom, y en la ESN se disparan las oscilaciones de la curva de validación, signo de que la red empieza a memorizar ruido.

Cuando los precios describen ascensos pronunciados seguidos de correcciones igual de rápidas, solo la KAN consigue sostener errores por debajo del 6 % de manera sistemática. Las demás redes tienden a quedarse cortas en los máximos y a reaccionar con retraso en los mínimos. En valores más estables, como Berkshire Hathaway, las diferencias se reducen: todas las arquitecturas rozan MAPE del 4 % - 6 % a diez años y el criterio para elegir pasa a ser el coste computacional.

KAN y LSTM acostumbran a estabilizar la pérdida de validación antes de diez épocas útiles. En ESN y LNN la validación baja al principio, pero después aparecen picos: en ESN esos picos coinciden con tramos donde la serie cambia de pendiente, y en LNN se prolongan durante decenas de épocas. No se observó sobreajuste grave en la KAN; en cambio, la combinación ESN y ventana larga genera validaciones crecientes, indicio de que termina aprendiendo variaciones aleatorias.

Si se suman todas las ejecuciones (diez compañías por tres horizontes, es decir 120 modelos), la KAN consume en torno a 55 min en una CPU estándar. LSTM duplica ese tiempo por su mayor número de pesos; LNN roza las cuatro horas;

y ESN se queda cerca de la media hora. Con un presupuesto de cálculo limitado la KAN ofrece la mejor relación precisión-tiempo.

En resumen, para precios diarios entre 2012 y 2025, la Kolmogorov-Arnold Network combina el menor error, la mayor estabilidad y el modelo más pequeño. LSTM es una alternativa sólida cuando la volatilidad no es extrema y se puede asumir un coste mayor. ESN sirve como solución rápida si se trabaja con series no muy largas y recursos escasos, mientras que LNN solo resulta prometedora en contextos de frecuencia muy alta, donde su dinámica continua podría dar ventaja. En la práctica, una sola KAN alimentada con pocas variables técnicas basta para seguir la tendencia principal; si además se quiere reaccionar a giros de muy corto plazo se podría añadir una LSTM como detector temprano y mantener una ESN ligera como respaldo.

5 Conclusiones

A lo largo del desarrollo de este trabajo se han cumplido satisfactoriamente los objetivos planteados inicialmente. Se ha realizado un análisis en profundidad de las bases teóricas de las redes neuronales seleccionadas: ESN, KAN, LSTM y LNN. Este estudio ha permitido entender claramente cómo funciona cada arquitectura, sus fortalezas específicas, y cómo abordan la predicción de series temporales financieras.

A partir de ese conocimiento, se implementaron versiones operativas de cada modelo utilizando los mismos datos históricos del índice S&P 500, siguiendo procedimientos de preprocesado y división idénticos para asegurar una comparación justa y fiable. Los experimentos realizados han sido consistentes, permitiendo evaluar con claridad el comportamiento individual de cada red y optimizar sus hiperparámetros a través del conjunto de validación.

Se ha empleado un conjunto sólido de métricas, como MSE, RMSE, MAE, R^2, MAPE y SMAPE, combinando herramientas estándar de scikit-learn y una versión propia de SMAPE, lo que aporta rigor y replicabilidad al estudio. Estas métricas han facilitado una evaluación transparente del rendimiento de cada arquitectura en términos prácticos, reflejando tanto su precisión predictiva como el coste computacional implicado.

Tras analizar críticamente los resultados, se ha confirmado claramente la superioridad del modelo KAN, especialmente en contextos de alta volatilidad, debido a su rapidez, precisión y sencillez estructural. La red LSTM se ha mostrado como una alternativa robusta, especialmente útil cuando las series presentan fluctuaciones más suaves. Por otro lado, los modelos ESN y LNN han mostrado un rendimiento menos consistente, aunque con potencial específico en contextos particulares, lo que invita a seguir investigando.

Finalmente, podemos concluir que el trabajo desarrollado no solo cumple con los objetivos específicos planteados inicialmente, sino que proporciona una base clara para futuras investigaciones, ampliando tanto la escala como la complejidad de los experimentos realizados.

5.1 Trabajo futuro

Una extensión lógica del presente trabajo sería realizar los mismos experimentos utilizando datos intradías. Por ejemplo, reconstruyendo las series temporales con intervalos de una hora o de cuatro horas se podría comprobar si los resultados obtenidos con datos diarios, donde la KAN fue superior seguida por la LSTM, se mantienen o cambian cuando el mercado genera mayor volatilidad y ruido. Este cambio obligaría a adaptar algunos parámetros como el tamaño de las secuencias o el escalado de datos, aunque no sería necesario modificar la arquitectura básica de las redes neuronales. Así, se podría analizar con claridad si las ventajas observadas con datos diarios se mantienen también cuando se predicen movimientos de corto plazo, o si alguno de los modelos se desempeña peor en condiciones de mayor incertidumbre.

Otra posibilidad interesante para ampliar la investigación sería desarrollar modelos que manejen datos de varias empresas simultáneamente. Hasta ahora, cada modelo se ha entrenado con datos individuales de una sola compañía, pero podría tener sentido utilizar una matriz multivariante con las cotizaciones de varias acciones, especialmente si pertenecen a sectores relacionados o están afectadas por los mismos factores económicos. Este planteamiento permitiría que los modelos aprovecharan correlaciones entre empresas, lo que podría mejorar considerablemente las predicciones. A nivel práctico, implicaría únicamente reorganizar los datos de entrada y salida para que el modelo prediga

varias acciones al mismo tiempo, sin necesitar modificaciones complejas en su estructura interna.

Por último, sería muy útil automatizar la actualización de los modelos. La idea sería crear un script que al final de cada día añada automáticamente la nueva información de mercado y vuelva a entrenar brevemente el modelo. Si este reentrenamiento logra reducir el error en los datos de validación, se mantendría esa versión mejorada del modelo. Si no es así, se mantendría la versión anterior. Este enfoque automatizado simplificaría significativamente el proceso de mantenimiento, limitando el trabajo manual al seguimiento ocasional del rendimiento de los modelos.

5.2 Análisis de impacto

El uso de un modelo que requiere muy pocos parámetros y tiempos de entrenamiento reducidos presenta beneficios evidentes para los Objetivos de Desarrollo Sostenible. El ahorro energético derivado de ciclos de cálculo breves y de la ausencia de hardware especializado contribuye de forma directa a la producción responsable y a la mitigación de emisiones (ODS 12 y ODS 13). Al reducir la barrera de entrada técnica y económica, se fomenta además la inclusión de actores con menor capacidad financiera, alineándose con la meta de reducción de desigualdades (ODS 10) y con la promoción de una innovación asequible (ODS 9).

La transparencia inherente a la estructura univariante de la KAN favorece la auditoría y la trazabilidad de las decisiones algorítmicas, aspecto relevante para consolidar instituciones sólidas y una gobernanza ética de la inteligencia artificial (ODS 16). No obstante, se reconoce el riesgo de que dichas mejoras sean aprovechadas por operadores de alta frecuencia cuyas prácticas puedan aumentar la asimetría informativa del mercado. Mitigar esta externalidad negativa exige difundir abiertamente el código y los conjuntos de hiperparámetros, así como establecer protocolos de revisión periódica que detecten sesgos y derivas del modelo.

En términos globales, la investigación confirma que es posible avanzar en la precisión de las predicciones bursátiles sin incurrir en costes energéticos ni computacionales elevados, siempre que se privilegien diseños frugales y se mantenga un compromiso real con la apertura y la supervisión responsable de la tecnología.

6 Bibliografía

[1] Macabacus, "Financial Forecasting with Time Series Analysis," Macabacus Blog, 2023. [Online]. Available: https://macabacus.com/blog/financial-forecasting-with-time-series-analysis

[2] A. Bouziane, S. Aloui, et al., "Deep learning for time series forecasting: a survey," J. Ambient Intell. Humaniz. Comput., Springer, 2025. [Online]. Available: https://link.springer.com/article/10.1007/s13042-025-02560-w

[3] Paradigma Digital, "¿Cómo predecir las ventas en el sector retail con series temporales?," Paradigma Digital, 2022. [Online]. Available: https://www.paradigmadigital.com/dev/predecir-ventas-retail-series-temporales-prophet/

[4] R. J. Hyndman and G. Athanasopoulos, Forecasting: Principles and Practice, 3rd ed. OTexts, 2021. [Online]. Available: https://otexts.com/fpp3/

[5] R. Ahmed and C. N. Kolachi, "Deep Learning-Based Stock Market Forecasting Techniques: A Survey," IEEE Access, vol. 9, pp. 151077-151104, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9532341

[6] MathWorks, "What is Time Series Analysis?," MathWorks, 2023. [Online]. Available: https://la.mathworks.com/discovery/time-series-analysis.html

[7] Futuros Trading LLC, "La última guía de trading con medias móviles," Futuros.com, 05 septiembre 2022. [Online]. Available: https://futuros.com/la-ultima-guia-de-trading-con-medias-moviles/

[8] K. Davey, "GARCH (p, q) Models for Time Series Analysis," QuantStart, 2023. [Online]. Available: https://www.quantstart.com/articles/Generalised-Autoregressive-Conditional-Heteroskedasticity-GARCH-p-q-Models-for-Time-Series-Analysis/

[9] Investopedia, "Trend Analysis Definition," Investopedia, 2023. [Online]. Available: https://www.investopedia.com/terms/t/trendanalysis.asp

[10] L. Álvarez, "Series temporales: Descomposición STL," AEDV Bookdown, 2023. [Online]. Available: https://bookdown.org/lalvarez_mat/AEDV/series-temporales.html#descomposici%C3%B3n-stl

[11] DataCamp, "Time Series Forecasting Tutorial," DataCamp, 2021. [Online]. Available: https://www.datacamp.com/tutorial/tutorial-time-series-forecasting

[12] Concur, "Análisis en tiempo real y su impacto en la toma de decisiones," SAP Concur, 2023. [Online]. Available: https://www.concur.com.mx/blog/article/analisis-en-tiempo-real

[13] A. Karpathy, "The Unreasonable Effectiveness of Recurrent Neural Networks," 2015. [Online]. Available: https://karpathy.github.io/2015/05/21/rnn-effectiveness/

[14] M. Lukosevicius, "A Practical Guide to Applying Echo State Networks," in Neural Networks: Tricks of the Trade, Springer, 2012, pp. 659-686. [Online]. Available: https://www.ai.rug.nl/minds/uploads/PracticalESN.pdf

[15] H. Jaeger, "The 'echo state' approach to analysing and training recurrent neural networks," GMD Technical Report 148, German National Research Center for Information Technology, 2001. [Online]. Available: https://www.researchgate.net/figure/The-architecture-of-standard-Echo-State-Network-ESN_fig1_283334340

[16] S. Moin and S. Karim, "Cryptocurrency Price Forecasting Using Echo State Networks," arXiv preprint, arXiv:2203.12391, 2022. [Online]. Available: https://arxiv.org/abs/2203.12391

[17] M. Tan and C. P. Lim, "An Overview of Reservoir Computing: Echo State Networks and Their Applications," Information, vol. 14, no. 3, pp. 157-172, 2023. [Online]. Available: https://www.mdpi.com/2078-2489/14/3/157

[18] A. N. Kolmogorov, "On the representation of continuous functions of many variables by superpositions of continuous functions of one variable and addition," Dokl. Akad. Nauk SSSR, vol. 114, pp. 953-956, 1957.[Online]. Available: https://cs.uwaterloo.ca/~y328yu/classics/Kolmogorov57.pdf

[19] M. Wang and C. Uhler, "Kolmogorov-Arnold Networks," arXiv preprint, arXiv:2304.00635, 2023. [Online]. Available: https://arxiv.org/abs/2304.00635

[20] Python's Gurus, "KAN (Kolmogorov Arnold Networks) - A Short Summary," Medium, 2023. [Online]. Available: https://medium.com/pythons-gurus/kan-kolmogorov-arnold-networks-a-short-summary-a1aef1336990

[21] C. Dong, L. Zheng y W. Chen, "Kolmogorov-Arnold Networks (KAN) for Time Series Classification and Robust Analysis," arXiv preprint arXiv:2408.07314, 2024. [Online]. Available: https://arxiv.org/abs/2408.07314

[22] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Comput., vol. 9, no. 8, pp. 1735-1780, Aug. 1997. [Online]. Available: https://www.bioinf.jku.at/publications/older/2604.pdf

[23] ConoceMachineLearning, "Por qué funcionan tan bien las LSTM en NLP frente a las redes estáticas," ConoceMachineLearning, 24 de julio de 2017. [Online]. Available: https://conocemachinelearning.wordpress.com/2017/07/24/por-que-funcionan-tan-bien-las-lstm-en-nlp-frente-a-las-redes-estaticas/

[24] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to Forget: Continual Prediction with LSTM," Neural Comput., vol. 12, no. 10, pp. 2451-2471, Oct. 2000. [Online]. Available: https://sferics.idsia.ch/pub/juergen/FgGates-NC.pdf

[25] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," arXiv preprint, arXiv:1412.6980, 2015. [Online]. Available: https://arxiv.org/abs/1412.6980

[26] D. M. Nelson, A. C. M. Pereira, and R. A. de Oliveira, "Stock market's price movement prediction with LSTM neural networks," in Proc. Int. Joint Conf. Neural Netw. (IJCNN), 2017. [Online]. Available: https://www.researchgate.net/publication/318329563_Stock_market's_price_movement_prediction_with_LSTM_neural_networks

[27] F. Karim, S. Majumdar, H. Darabi, and S. Chen, "LSTM Fully Convolutional Networks for Time Series Classification," IEEE Access, vol. 6, pp. 1662-1669, 2018. [Online]. Available: https://doi.org/10.1109/ACCESS.2017.2779939

[28] R. Hasani, M. Lechner, A. Amini, D. Rus, and R. Grosu, "Liquid time-constant networks," arXiv preprint, arXiv:2006.04439, 2021. [Online]. Available: https://arxiv.org/abs/2006.04439

[29] R. Khandelwal, "Liquid Neural Network: Pioneering Adaptive Intelligence," Medium, 25 de noviembre de 2024. [Online]. Available:

https://arshren.medium.com/liquid-neural-network-pioneering-adaptive-intelligence-d00313f7197e

[30] M. Lechner et al., “Neural Circuit Policies Enabling Auditable Autonomy,” Nature Machine Intelligence, vol. 2, no. 10, pp. 642-652, 2020. [Online]. Available: https://repositum.tuwien.at/handle/20.500.12708/141225

[31] R. Hasani, M. Lechner, D. Rus, and R. Grosu, "Closed-form continuous-time neural networks," arXiv preprint, arXiv:2106.13898, 2022. [Online]. Available: https://arxiv.org/abs/2106.13898

[32] J. Woo, S.-H. Kim, H. Kim, and K. Han, “Characterization of the neuronal and network dynamics of liquid state machines,” Physica A: Statistical Mechanics and Its Applications, vol. 633, Art. 129334, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378437123008890

[33] R. Hasani et al., "Liquid Time-constant networks for real-time learning in robotics and beyond," arXiv preprint, arXiv:2107.03437, 2021. [Online]. Available: https://arxiv.org/abs/2107.03437

[34] P. Martí Sanahuja, "Métricas de evaluación de rendimiento para predicciones de series temporales," 2023. [Online]. Available: https://polmartisanahuja.com/metricas-de-evaluacion-de-rendimiento-para-predicciones-de-series-temporales/

[35] M. Bayraktar, “What is SMAPE?”, 2023. [Online]. Available: https://medium.com/@mertbayraktarxd/what-is-smape-2cf605831feb

[36] Analytics Vidhya, “Introduction to Cost Function,” 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/03/data-science-101-introduction-to-cost-function/

[37] scikit-learn, “sklearn.metrics.mean_squared_error,” scikit-learn documentation, 2024. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_squared_error.html

[38] ML Cheatsheet, “Adam Optimizer,” 2017. [Online]. Available: https://ml-cheatsheet.readthedocs.io/en/latest/optimizers.html#adam

[39] Scikit-learn, “sklearn.preprocessing.MinMaxScaler,” Scikit-learn, 2024. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

[40] S. Nalcin, “StandardScaler vs. MinMaxScaler vs. RobustScaler,” Medium, 2022. [Online]. Available: https://medium.com/@onersarpnalcin/standardscaler-vs-minmaxscaler-vs-robustscaler-which-one-to-use-for-your-next-ml-project-ae5b44f571b9

[41] D. Masters and C. Luschi, “Revisiting Small Batch Training for Deep Neural Networks,” arXiv preprint, arXiv:1804.07612, 2018. [Online]. Available: https://arxiv.org/abs/1804.07612

[42] E. Hoffer, I. Hubara, and D. Soudry, “Train longer, generalize better: closing the generalization gap in large batch training of neural networks,” arXiv preprint, arXiv:1705.08741, 2017. [Online]. Available: https://arxiv.org/abs/1705.08741

[43] Y. Bengio, “Practical Recommendations for Gradient-Based Training of Deep Architectures,” arXiv preprint, arXiv:1206.5533, 2012. [Online]. Available: https://arxiv.org/abs/1206.5533

[44] PyTorch, “ReduceLROnPlateau,” PyTorch Documentation, 2024. [Online]. Available: https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.ReduceLROnPlateau.html

[45] IBM, “Backpropagation,” IBM, 2023. [Online]. Available: https://www.ibm.com/topics/backpropagation

[46] IBM Cloud Education, “Gradient Descent,” IBM, 2023. [Online]. Available: https://www.ibm.com/cloud/learn/gradient-descent

[47] G. D. Santos et al., "Machine learning applied in the stock market through the Moving Average Convergence Divergence (MACD) indicator," 2020. [Online]. Available: https://www.researchgate.net/publication/347846403

[48] Forex Strategies Work, “Ultimate Guide to the MACD Indicator,” Forex Strategies Work, [Online]. Available: https://www.forexstrategieswork.com/ultimate-guide-to-the-macd-indicator/

[49] L. Prechelt, “Early Stopping — But When?,” in Neural Networks: Tricks of the Trade, Springer, 1998, pp. 55-69. [Online]. Available: https://page.mi.fu-berlin.de/prechelt/Biblio/stop_tricks1997.pdf

[50] Scikit-learn, “sklearn.model_selection.train_test_split,” Scikit-learn documentation, 2024. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

[51] Investopedia, “Top 25 Stocks in the S&P 500 by Index Weight for May 2025,” Investopedia, 2025. [Online]. Available: https://www.investopedia.com/the-best-25-sp500-stocks-8778635

[52] Yahoo Finance, “Yahoo Finance - Stock Market Live, Quotes, Business & Finance News,” Yahoo Finance, 2025. [Online]. Available: https://finance.yahoo.com/

[53] S. Vasanth, S. Geetha and P. K. B. P. Kumar, "Stock Closing Price Prediction using Machine Learning Techniques," *Procedia Computer Science*, vol. 172, pp. 1045-1053, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050920307924

[54] S. Salinas, M. Sadeghi and T. Badger, "The choice of scaling technique matters for classification performance," *arXiv preprint*, arXiv:2212.12343, 2022. [Online]. Available: https://arxiv.org/pdf/2212.12343

[55] S. Onardo, “pytorch-esn,” GitHub repository, 2021. [Online]. Available: https://github.com/stefanonardo/pytorch-esn

[56] B. Aleitan, “efficient-kan,” GitHub repository, 2023. [Online]. Available: https://github.com/Blealtan/efficient-kan

[57] PyTorch, “examples,” GitHub repository, 2025. [Online]. Available: https://github.com/pytorch/examples

[58] A. Sugihartono, D. S. Wijaya y A. Rizal, “LSTM Network Hyperparameter Optimization for Stock Price Prediction Using the Optuna Framework,” eprints.uad.ac.id, 2023. [Online]. Available: https://eprints.uad.ac.id/41357/1/3-LSTM%20Network%20Hyperparameter%20Optimization%20for%20Stock%20Price%20Prediction%20Using%20the%20Optuna%20Framework.pdf

[59] R. Hasani, “liquid_time_constant_networks,” GitHub repository, 2021. [Online]. Available: https://github.com/raminmh/liquid_time_constant_networks

[60] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," European Journal of Operational Research, vol. 270, no. 2, pp. 654-669, 2018. [Online]. Available: https://doi.org/10.1016/j.ejor.2017.11.054

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
	Fecha/Hora	Tue Jun 03 21:52:03 CEST 2025
	Emisor del Certificado	EMAILADDRESS=camanager@etsiinf.upm.es, CN=CA ETS Ingenieros Informaticos, O=ETS Ingenieros Informaticos - UPM, C=ES
	Numero de Serie	561
	Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)