Trabajo Fin de Grado
Ingeniería Electrónica, Robótica y Mecatrónica

Artificial intelligence techniques and their application to time series forecast

Autor: Antonio Luis González Hernández

Tutor: Juan Antonio Becerra González

Dpto. Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2024

Trabajo Fin de Grado
Ingeniería Electrónica, Robótica y Mecatrónica

Artificial intelligence techniques and their application to time series forecast

Autor:
Antonio Luis González Hernández

Tutor:
Juan Antonio Becerra González
Profesor Titular

Dpto. Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2024

Trabajo Fin de Grado:	Artificial intelligence techniques and their application to time series forecast

Autor:	Antonio Luis González Hernández
Tutor:	Juan Antonio Becerra González

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Índice Abreviado

Contents

1 History and State of the Art

1.1 Evolution of Methods for Time Series Forecasting

Time series forecasting is a fundamental discipline in various fields such as economics, meteorology, engineering, and social sciences. Throughout history, this discipline has evolved significantly, developing increasingly sophisticated methods to improve the accuracy and reliability of predictions. In this chapter, we will explore the evolution of time series forecasting methods, from the most traditional approaches to the most modern ones based on machine learning techniques and neural networks.

1.1.1 Traditional Methods

Traditional time series forecasting methods have been the cornerstone of predictive analytics for decades. These methods typically rely on well-established statistical principles to analyze historical data and make future predictions based on observed patterns and relationships. A common feature of traditional methods is their emphasis on linear relationships and the assumption that historical patterns will persist into the future. They are grounded in mathematical theories and formulas, providing a structured and straightforward approach to forecasting. This simplicity contributes to their ease of interpretation, making it easier for practitioners to understand and communicate results. Moreover, traditional methods often assume that the time series data is stationary, meaning its statistical properties, such as mean and variance, do not change over time. As a result, techniques within these methods frequently involve transforming non-stationary data into a stationary format. They focus on capturing dependencies between current and past values of the time series, often through lagged values, and incorporate mechanisms to handle seasonality and trends, adjusting for periodic fluctuations and long-term movements in the data. Parameter estimation is a key aspect, with these methods typically involving the calculation of a set number of parameters based on historical data.

Autoregressive and Moving Average Models (AR, MA, ARMA)

The first time series forecasting models included autoregressive (AR) and moving average (MA) models. These models are linear and are based on the assumption that the time series can be explained by a linear combination of its past values and an error term. The ARMA model combines both approaches to provide a more robust tool for time series analysis.

Autoregressive Models (AR) of order p (AR(p)) uses the p previous values of the time series to make predictions. The general formula for an AR(p) model is:

$$X_t = c + \sum_{i=1}^{p} \phi_i X_{t-i} + \varepsilon_t$$

where:

- X_t is the value of the time series at time t.
- c is a constant.
- ϕ_i are the model coefficients.
- ε_t is the error term at time t.

Moving Average Models (MA) of order q (MA(q)) uses q previous error terms to make predictions. The general formula for an MA(q) model is:

$$X_t = \mu + \varepsilon_t + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j}$$

where:

- μ is the mean of the time series.
- θ_j are the coefficients of the model.
- ε_t is the error term at time t.

Autoregressive Moving Average (ARMA) models combines both approaches to capture both autoregressive and moving average dependence. The general formula for an ARMA(p,q) model is:

$$X_t = c + \sum_{i=1}^{p} \phi_i X_{t-i} + \varepsilon_t + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j}$$

where:

- X_t is the value of the time series at time t.
- c is a constant.
- ϕ_i are the autoregressive coefficients.
- θ_j are the moving average coefficients.
- ε_t is the error term in time t.

Autoregressive Integrated Moving Average (ARIMA) and SARIMA Models

For non-stationary time series, integrated autoregressive moving average (ARIMA) models were introduced. These models include a differencing term to handle non-stationarity. The SARIMA model extends ARIMA by including seasonal components, providing a more robust tool for the analysis of time series with seasonal patterns.

Autoregressive Moving Average Integrated Models (ARIMA) combines the autoregressive (AR), differencing (I), and moving average (MA) components. The general formula for an ARIMA(p,d,q) model is:

$$\Delta^d X_t = c + \sum_{i=1}^{p} \phi_i \Delta^d X_{t-i} + \varepsilon_t + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j}$$

where:

- X_t is the value of the time series at time t.
- Δ^d represents the order differentiation d.
- c is a constant.
- ϕ_i are the autoregressive coefficients.
- θ_j are the moving average coefficients.
- ε_t is the error term in time t.

The SARIMA model extends the ARIMA model by including seasonal components. The general formula for a SARIMA(p,d,q)(P,D,Q,m) model is:

$$\Delta^d \Delta_m^D X_t = c + \sum_{i=1}^{p} \phi_i \Delta^d \Delta_m^D X_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \sum_{k=1}^{P} \Phi_k \Delta_m^D X_{t-km} + \sum_{l=1}^{Q} \Theta_l \varepsilon_{t-lm}$$

where:

- X_t is the value of the time series at time t.

- Δ^d represents the differencing of order d.
- Δ_m^D represents the seasonal differencing of order D with periodicity m.
- c is a constant.
- ϕ_i are the autoregressive coefficients.
- θ_j are the moving average coefficients.
- Φ_k are the seasonal autoregressive coefficients.
- Θ_l are the seasonal moving average coefficients.
- ε_t is the error term at time t.

SARIMAX models

SARIMAX models (Seasonal ARIMA with eXogenous regressors) are an extension of SARIMA models that allow the inclusion of exogenous variables in time series analysis. These exogenous variables are external factors that can influence the time series and, by incorporating them, the accuracy of the predictions is improved. The general structure of a SARIMAX model can be expressed as:

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q} + \beta_1 X_{1,t} + \beta_2 X_{2,t} + \cdots + \beta_k X_{k,t} + \varepsilon_t$$

where:

- Y_t is the value of the time series at time t.
- ϕ_i are the autoregressive coefficients.
- θ_i are the moving average coefficients.
- ε_t is the error term at time t.
- $X_{i,t}$ are the exogenous variables at time t.
- β_i are the coefficients of the exogenous variables.

VARMA and VARMAX Models

VARMA (Vector AutoRegressive Moving Average) models are a multivariate extension of ARMA models, suitable for analyzing time series with multiple interrelated variables. VARMAX (Vector AutoRegressive Moving Average with eXogenous regressors) models include additional exogenous variables, allowing the incorporation of external information that can influence the time series. VARMA models are useful when you have multiple time series that can influence each other. The general structure of a VARMA(p, q) model can be expressed as:

$$Y_t = \Phi_1 Y_{t-1} + \Phi_2 Y_{t-2} + \cdots + \Phi_p Y_{t-p} + \Theta_1 \varepsilon_{t-1} + \Theta_2 \varepsilon_{t-2} + \cdots + \Theta_q \varepsilon_{t-q} + \varepsilon_t$$

where:

- Y_t is a vector of time series values at time t.
- Φ_i are autoregressive coefficient matrices.
- Θ_i are moving average coefficient matrices.
- ε_t is a vector of error terms at time t.

Simple Exponential Smoothing (SES) and Holt-Winters Exponential Smoothing (HWES)

Simple Exponential Smoothing (SES) and Holt-Winters Exponential Smoothing (HWES) are time series forecasting methods that use exponential smoothing techniques to capture patterns in the data. These methods are especially useful for time series with trend and seasonality components.

Simple Exponential Smoothing (SES) is a forecasting technique that assigns exponentially decreasing weights to past observations. The general formula for SES is:

$$\hat{Y}_{t+1} = \alpha Y_t + (1-\alpha)\hat{Y}_t$$

where:

- $\hat{Y}_{t+1}$ is the forecast for time $t+1$.
- Y_t is the observed value at time t.
- $\hat{Y}_t$ is the forecast for time t.
- α is the smoothing parameter $(0 < \alpha < 1)$.

Holt-Winters Exponential Smoothing (HWES) is an extension of SES that incorporates trend and seasonality components. There are two main variants of the Holt-Winters method: additive and multiplicative.

The Additive Method is suitable for time series with constant seasonality over time. The equations of the additive method are:

Level:

$$L_t = \alpha(Y_t - S_{t-m}) + (1-\alpha)(L_{t-1} + T_{t-1})$$

Trend:

$$T_t = \beta(L_t - L_{t-1}) + (1-\beta)T_{t-1}$$

Seasonality:

$$S_t = \gamma(Y_t - L_t) + (1-\gamma)S_{t-m}$$

Forecast:

$$\hat{Y}_{t+h} = L_t + hT_t + S_{t+h-m(k+1)}$$

where:

- L_t is the level at time t.
- T_t is the trend at time t.
- S_t is the seasonal component at time t.
- m is the number of periods in a season.
- h is the forecast horizon.
- α, β, and γ are the smoothing parameters.

The Multiplicative Method is suitable for time series with seasonality that varies proportionally with the level of the series. The equations of the multiplicative method are:

Level:

$$L_t = \alpha\frac{Y_t}{S_{t-m}} + (1-\alpha)(L_{t-1} + T_{t-1})$$

Trend:

$$T_t = \beta(L_t - L_{t-1}) + (1-\beta)T_{t-1}$$

Seasonality:

$$S_t = \gamma\frac{Y_t}{L_t} + (1-\gamma)S_{t-m}$$

Forecast:

$$\hat{Y}_{t+h} = (L_t + hT_t)S_{t+h-m(k+1)}$$

ARCH and GARCH Models

ARCH (Autoregressive Conditional Heteroskedasticity) and GARCH (Generalized ARCH) models are widely used in financial time series analysis, where the volatility of the data can change over time. These models are especially useful for capturing conditional variability in financial data, such as stock prices or exchange rates.

The ARCH model, proposed by Robert Engle in 1982, models the conditional variance of a time series as a function of past errors. The general structure of an ARCH(p) model can be expressed as follows:

$$Y_t = \mu + \varepsilon_t$$

$$\varepsilon_t = \sigma_t Z_t$$

$$\sigma_t^2 = \alpha_0 + \alpha_1\varepsilon_{t-1}^2 + \alpha_2\varepsilon_{t-2}^2 + \cdots + \alpha_p\varepsilon_{t-p}^2$$

where:

- Y_t is the value of the time series at time t.
- μ is the mean of the time series.
- ε_t is the error term at time t.
- σ_t^2 is the conditional variance at time t.
- Z_t is a random variable with standard normal distribution.
- $\alpha_0, \alpha_1, \ldots, \alpha_p$ are the parameters of the model.

The GARCH model, developed by Tim Bollerslev in 1986, is an extension of the ARCH model that allows for a more parsimonious and flexible representation of conditional variance. Instead of relying solely on past errors, the GARCH model also incorporates past conditional variance. This model is particularly useful because it can capture persistence in volatility, meaning that shocks to conditional variance can have long-lasting effects. This is especially relevant in financial analysis, where periods of high volatility tend to cluster.

The general structure of a GARCH(p, q) model can be expressed as:

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{p} \alpha_i \varepsilon_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2$$

where:

- σ_t^2 is the conditional variance at time t.
- α_0 is a constant.
- α_i are the coefficients of the ARCH terms.
- β_j are the coefficients of the GARCH terms.
- p is the number of lags of the ARCH terms.
- q is the number of lags of the GARCH terms.

1.1.2 Modern Methods

Modern time series forecasting methods leverage advancements in machine learning and computational power to address the complexities of contemporary data. These approaches are designed to handle large datasets and uncover intricate patterns that traditional methods might overlook. One of the defining features of modern methods is their ability to model non-linear relationships, capturing complex interactions within the data that traditional linear models might miss. They are highly flexible and adaptable, accommodating various types of data structures and managing non-stationary series without requiring explicit transformations.

Modern methods rely heavily on data-driven techniques, utilizing vast amounts of data to train models and improve predictive accuracy. Sophisticated feature engineering is often employed to extract relevant patterns and trends from raw data, enhancing model performance. Additionally, these methods can handle high-dimensional datasets with multiple variables, effectively capturing interdependencies and interactions. Many modern approaches use ensemble learning techniques, combining multiple models to improve robustness and accuracy. They are also designed to scale efficiently with the growing size of datasets, leveraging parallel processing and advanced computational resources. Furthermore, advanced optimization algorithms are used to fine-tune model parameters, achieving optimal performance through iterative learning processes.

Recurrent Neural Networks (RNNs) and LSTMs

Recurrent Neural Networks (RNN) and their Long Short-Term Memory (LSTM) variants have revolutionized the field of time series forecasting. These deep learning techniques are capable of capturing long-term dependencies in data, which makes them especially useful for this type of task.

RNNs were introduced in the 1980s as an extension of traditional neural networks, designed to process sequences of data. Unlike feedforward networks, RNNs have recurrent connections that allow information to persist. This means that they can use their internal memory to process sequences of inputs, which is crucial for tasks where temporal context is important.

RNNs are used in a variety of time series applications, such as stock price forecasting, energy demand, and weather data analysis. However, traditional RNNs have limitations when it comes to capturing long-term dependencies due to the gradient fading problem.

To overcome the limitations of traditional RNNs, Sepp Hochreiter and Jürgen Schmidhuber proposed **LSTMs** in 1997[5]. LSTMs are a variant of RNNs that include a memory architecture designed to handle long-term dependencies more efficiently. The key to LSTMs is their structure of memory cells, which can maintain and update information over many time steps.

LSTMs are composed of memory cells that have three gates: the input gate, the forget gate, and the output gate. These gates control the flow of information in and out of the memory cell, allowing the network to remember or ignore information as needed.

LSTMs have proven to be extremely effective in time series forecasting. They are used in applications such as traffic forecasting, financial data anomaly detection, and sales forecasting. Thanks to their ability to handle long-term dependencies, LSTMs outperform traditional RNNs in many time series tasks.

Residual Networks (ResNet) and Fully Convolutional Networks (FCN)

Convolutional neural networks (CNNs) have revolutionized the field of machine learning, especially in computer vision tasks. However, their applications have been extended to other domains, including time series forecasting. Among the most prominent variants of CNNs are Residual Networks (ResNet) and Fully Convolutional Networks (FCN).

Both ResNets and FCNs have been successfully applied in time series forecasting. ResNets, with their ability to handle deep networks, are ideal for modeling complex and non-linear relationships in multivariate data. On the other hand, FCNs, with their focus on full convolution, allow detailed patterns to be captured and accurate predictions to be made in dense time series.

These architectures have proven to be powerful tools in time series analysis and forecasting, offering significant improvements in accuracy and efficiency compared to traditional methods such as ARIMA and SARIMAX.

Residual Networks (ResNet), introduced by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun in 2015, emerged as a solution to the problem of degradation in deep neural networks. This problem refers to the decrease in accuracy as more layers are added to a neural network. ResNets address this challenge by introducing skip connections that allow information to flow directly through the network, skipping one or more intermediate layers.

These skip connections facilitate the training of very deep networks, allowing the construction of models with hundreds of layers without significant degradation in performance. ResNets have proven effective in a wide range of applications, including time series forecasting, where their ability to handle complex, multivariate data is particularly valuable.

Fully Convolutional Networks (FCNs)[12] were proposed by Jonathan Long, Evan Shelhamer, and Trevor Darrell in 2015 as an extension of traditional CNNs for semantic segmentation tasks. Unlike conventional CNNs, FCNs replace fully connected layers with convolutional layers, allowing the network to process inputs of arbitrary size and produce outputs of corresponding size.

This architecture has been adapted for time series forecasting, taking advantage of its ability to capture spatial and temporal patterns in the data. FCNs are especially useful in scenarios where dense and detailed forecasting is required, such as in the case of multivariate time series.

ROCKET

ROCKET (Random Convolutional Kernel Transform)[4] is an innovative and recent method in the field of time series analysis. It was introduced by Angus Dempster, François Petitjean, and Geoffrey I. Webb in 2019. This method is notable for its ability to apply random convolutional transformations to time series data, extracting useful features that can be used by simple linear classifiers.

The fundamental principle of ROCKET is the use of a large number of one-dimensional (1-D) convolutional kernels that are randomly initialized. These kernels are applied to the time series to generate a transformed representation of the data. Unlike other methods that require extensive training of the kernels, in ROCKET the kernels are not trained, which significantly reduces the computational time.

Once the features have been extracted using convolutional transformations, linear classifiers, such as the RidgeClassifier, are used to perform time series classification. This approach allows ROCKET to be exceptionally fast and efficient, achieving high accuracy results at a fraction of the computational cost of other advanced methods.

ROCKET has proven to be highly effective in a variety of time series classification tasks. Its ability to handle large volumes of data and its computational efficiency make it ideal for real-time applications and in scenarios where computational resources are limited. In addition, its simplicity and speed allow it to be deployed on a wide range of platforms and environments.

Matrix Factorization with DTW

Matrix factorization is a fundamental technique in linear algebra that decomposes a matrix into products of simpler matrices. When combined with Dynamic Time Warping (DTW), a time alignment technique, it becomes a powerful tool for multivariate time series analysis.

Dynamic Time Warping (DTW)[1] is an algorithm that allows two time series to be aligned nonlinearly by adjusting for differences in the speed of the sequences. This is particularly useful in time series analysis where events may occur at different times but follow a similar pattern. Combining DTW with matrix factorization allows for improved temporal alignment between different series, facilitating the identification of common patterns.

Matrix factorization, on the other hand, decomposes a matrix into simpler matrix products, which facilitates data analysis and manipulation. By applying DTW in the context of matrix factorization, more accurate and aligned representations of multivariate time series can be obtained.

Matrix factorization combined with DTW represents an advanced and efficient technique for multivariate time series analysis. Its ability to improve time alignment and simplify data makes it a valuable tool for researchers and practitioners in various fields.

XGBoost

XGBoost[3], short for eXtreme Gradient Boosting, is a decision tree-based boosting algorithm that has gained popularity in the machine learning field due to its efficiency and accuracy. It was developed by Tianqi Chen and Carlos Guestrin in 2016. This algorithm has proven to be extremely effective for time series forecasting, especially because of its ability to handle tabular data efficiently.

XGBoost is an optimized implementation of the gradient boosting algorithm, which combines multiple weak decision trees to form a strong model. The boosting process involves the sequential construction of trees, where each new tree attempts to correct errors made by previous trees. This is achieved by minimizing a loss function, which allows the model to iteratively improve its accuracy.

One of the distinguishing features of XGBoost is its ability to handle tabular data efficiently. Tabular data, consisting of rows and columns, is common in many real-world applications, such as finance, medicine, and marketing. XGBoost excels in this type of data due to its ability to handle heterogeneous features and its computational efficiency.

Prophet

Prophet[16] is an open source tool developed by Facebook's Data Science team in 2017. It is designed for time series forecasting and is based on an additive component model, which facilitates the inclusion of trends and seasonalities automatically.

Prophet uses an additive model where the time series observations are decomposed into three main components: trend, seasonality, and holidays. The basic formula of the model is:

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t \tag{1.1}$$

Where:

- **Trend** $g(t)$: Captures the long-term evolution of the time series. It can be linear or logarithmic, and Prophet allows the inclusion of change points to model sudden changes in the trend.
- **Seasonality** $s(t)$: Models recurring patterns over different time periods, such as annual, weekly, or daily.
- **Holidays** $h(t)$: Allows the inclusion of effects of specific holidays that may influence the time series.
- **Error** ε_t: Represents noise or variations not explained by the previous components.

One of the advantages of Prophet is its ability to handle data with multiple and non-linear seasonalities, as well as its robustness to missing data and outliers.

Prophet represents a powerful and accessible tool for time series forecasting. Its approach based on an additive component model and its ability to handle complex data make it a preferred choice for analysts and data scientists worldwide.

TS-Chief and HIVE-COTE

TS-Chief[14] and HIVE-COTE[10] are advanced time series classification techniques based on ensemble learning. These techniques combine multiple classifiers to improve the accuracy and robustness of predictions by leveraging the strengths of different approaches.

TS-Chief is a time series classification technique that uses an approach based on decision trees and time series transformations. This method stands out for its ability to handle multivariate data and capture complex patterns in the time series. TSChef applies specific transformations to the time series to extract relevant features, which are then used by an ensemble of classifiers to make accurate predictions.

HIVE-COTE (Hierarchical Vote Collective of Transformation-based Ensembles) is a heterogeneous meta-assembly for time series classification. It was introduced by Anthony Bagnall and his team in 2016 and has been continuously updated to improve its performance. HIVE-COTE combines classifiers from multiple domains, including phase-independent shapelets, bag-of-words based dictionaries, and phase-dependent intervals. This combination allows HIVE-COTE to capture a wide variety of patterns in the time series, resulting in high accuracy and robustness in predictions.

1.2 Evolution of Transformers as a Method for Time Series Forecasting

The advent of Transformer models has revolutionized various fields of artificial intelligence, particularly natural language processing (NLP). Introduced by Vaswani et al. in 2017, the original Transformer architecture demonstrated unprecedented capabilities in handling sequential data, leading to significant advancements in machine translation, text generation, and more. However, the potential of Transformers extends far beyond NLP, finding promising applications in the domain of time series forecasting. Traditional methods for time series forecasting, such as ARIMA and exponential smoothing, have been effective but often struggle with capturing complex patterns and long-range dependencies in data. The application of Transformers to time series forecasting began to gain traction around 2019, with researchers exploring how the self-attention mechanism could be leveraged to model temporal dependencies more effectively. Since then, a variety of Transformer-based models have been proposed, each addressing specific challenges associated with time series data, such as scalability, interpretability, and handling of seasonality and trends. Much of this information is compiled in the article "Transformers in Time Series: A Survey"[19], which provides a comprehensive overview of the evolution of Transformers in time series analysis.

This section delves into the evolution of Transformers in the context of time series forecasting. We will explore the chronological development of key Transformer variants, highlighting their unique contributions and innovations. From the Temporal Fusion Transformer (TFT) to recent advancements like Crossformer and PatchTST, this section provides a comprehensive overview of how Transformers have been adapted and enhanced to meet the demands of time series forecasting.

1.2.1 Introduction of Transformers (2017)

Introduced by Vaswani et al.[17], the original Transformer architecture was primarily designed for natural language processing (NLP) tasks. It revolutionized the field by using a self-attention mechanism, which allowed the model to weigh the importance of different words in a sentence, regardless of their position. This architecture enabled parallel processing of data, significantly improving efficiency and performance in tasks like machine translation and text generation.

1.2.2 Application to Time Series (2019)

Temporal Fusion Transformer (TFT)
Proposed by Lim et al.[9], the Temporal Fusion Transformer was designed specifically for interpretable multi-horizon time series forecasting. TFT integrates both static and time-varying covariates, allowing it to

handle complex temporal patterns and provide insights into the importance of different features over time. Its interpretability makes it particularly useful in domains where understanding the model's decision-making process is crucial.

Informer
Introduced by Zhou et al.[24], the Informer model aimed to improve the efficiency and scalability of Transformers for long sequence time-series forecasting. It introduced a ProbSparse self-attention mechanism, which reduces the computational complexity by focusing on the most informative parts of the input sequence. This makes Informer more suitable for handling long sequences of time series data, where traditional Transformers would be computationally expensive.

1.2.3 Enhancements and New Variants (2020)

LogTrans
Developed by Li et al.[8], LogTrans focuses on capturing long-term dependencies in time series data. It introduces a logarithmic self-attention mechanism, which scales the attention scores logarithmically with the distance between elements in the sequence. This allows LogTrans to effectively model long-range dependencies without the quadratic complexity of standard self-attention, making it more efficient for long-term forecasting.

Reformer
Proposed by Kitaev et al.[7], the Reformer model aims to reduce the computational complexity of Transformers. It achieves this by using locality-sensitive hashing (LSH) to approximate the self-attention mechanism, significantly reducing the memory and computational requirements. Reformer also introduces reversible layers, which further decrease memory usage by allowing intermediate activations to be recomputed during backpropagation. This makes Reformer highly efficient and scalable for large-scale time series forecasting tasks.

1.2.4 Further Innovations (2021 - 2022)

Autoformer
Introduced by Wu et al.[21], Autoformer is designed for the automatic decomposition of time series into trend and seasonal components. This model leverages a decomposition block that separates the input time series into trend and seasonal parts, allowing the model to handle these components independently. By focusing on these distinct aspects of the data, Autoformer can achieve more accurate and interpretable forecasts, particularly in scenarios where understanding the underlying patterns is crucial.

FEDformer
Proposed by Zhou et al.[25], FEDformer (Frequency Enhanced Decomposition Transformer) focuses on frequency domain decomposition for time series forecasting. This model applies a Fourier transform to decompose the time series into different frequency components, which are then processed separately by the Transformer. By working in the frequency domain, FEDformer can capture periodic patterns more effectively and improve the accuracy of forecasts, especially for data with strong seasonal or cyclical behavior.

Pyraformer
Developed by Liu et al.[11], Pyraformer aims to improve the efficiency of Transformers for long sequence time series forecasting. It introduces a pyramidal attention mechanism that progressively reduces the sequence length at each layer, focusing on the most relevant parts of the input. This hierarchical approach allows Pyraformer to handle long sequences more efficiently, reducing computational complexity while maintaining high forecasting accuracy.

ETSformer Introduced by Zhou et al.[20], ETSformer combines exponential smoothing with Transformer architecture to better handle seasonality and trend in time series data. By integrating exponential smoothing, which is effective for capturing trends and seasonal patterns, with the powerful self-attention mechanism of Transformers, ETSformer can provide more accurate and robust forecasts. This hybrid approach leverages the strengths of both methods, making it particularly useful for time series with pronounced seasonal and trend components.

1.2.5 Recent Advances (2023 - 2024)

Crossformer Proposed by Zhang et al.[18], Crossformer focuses on cross-dimension dependency modeling for multivariate time series. This model introduces a cross-dimension attention mechanism that captures dependencies between different dimensions of the data, allowing for more accurate and holistic forecasting in multivariate settings. By effectively modeling the interactions between various time series, Crossformer enhances the predictive performance for complex datasets.

PatchTST Developed by Nie et al.[13] , PatchTST leverages patch-based input representations for time series forecasting. This approach divides the input time series into smaller patches, which are then processed independently before being aggregated. This patch-based method allows the model to capture local patterns more effectively and improves scalability for long sequences. PatchTST is particularly useful for handling large and complex time series data.

CARD Developed by Xue et al.[22], Channel Aligned Robust Blend Transformer (CARD) focuses on aligning and blending information across different channels of multivariate time series data. This model introduces a channel alignment mechanism that ensures consistent and robust integration of information from various channels, enhancing the model's ability to capture intricate dependencies and improve forecasting accuracy.

SageFormer Introduced by Zhang et al.[23]SageFormer is a Series-Aware Framework designed for long-term multivariate time series forecasting. This model incorporates series-aware attention mechanisms that specifically account for the unique characteristics of each time series in the dataset. By tailoring the attention mechanism to the properties of individual series, SageFormer achieves more accurate and reliable long-term forecasts.

InParformer Developed by Cao et al.[2], InParformer (Interactive Parallel Attention Transformer) introduces evolutionary decomposition with interactive parallel attention for long-term time series forecasting. This model decomposes the input time series into multiple components and applies parallel attention mechanisms to each component. The interactive nature of the attention mechanisms allows for better integration of information across components, leading to improved forecasting performance.

Stecformer Proposed by Sun et al.[15] Stecformer (Spatio-temporal Encoding Cascaded Transformer) is designed for multivariate long-term time series forecasting. This model employs spatio-temporal encoding to capture both spatial and temporal dependencies in the data. The cascaded architecture allows for hierarchical processing of information, enhancing the model's ability to handle complex multivariate time series and produce accurate long-term forecasts.

SAMformer Designed by Ilbert et al.[6], SAMformer (Sharpness-Aware Minimization and Channel-Wise Attention Transformer) aims to unlock the potential of Transformers in time series forecasting by incorporating sharpness-aware minimization and channel-wise attention mechanisms. Sharpness-aware minimization helps in optimizing the model's parameters for better generalization, while channel-wise attention focuses on capturing the importance of different channels in the data. This combination results in more robust and accurate forecasts.

List of Figures

List of Tables

List of Codes

Bibliography

[1] Karl Bringmann, Nick Fischer, Ivor van der Hoog, Evangelos Kipouridis, Tomasz Kociumaka, and Eva Rotenberg, *Dynamic dynamic time warping*, 2023.

[2] Haizhou Cao, Zhenhao Huang, Tiechui Yao, Jue Wang, Hui He, and Yangang Wang, *Inparformer: Evolutionary decomposition transformers with interactive parallel attention for long-term time series forecasting*, Proceedings of the AAAI Conference on Artificial Intelligence **37** (2023), no. 6, 6906–6915.

[3] Tianqi Chen and Carlos Guestrin, *Xgboost: A scalable tree boosting system*, Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, ACM, August 2016.

[4] Angus Dempster, François Petitjean, and Geoffrey I. Webb, *Rocket: exceptionally fast and accurate time series classification using random convolutional kernels*, Data Mining and Knowledge Discovery **34** (2020), no. 5, 1454–1495.

[5] Sepp Hochreiter and Jürgen Schmidhuber, *Long short-term memory*, Neural computation **9** (1997), 1735–80.

[6] Romain Ilbert, Ambroise Odonnat, Vasilii Feofanov, Aladin Virmaux, Giuseppe Paolo, Themis Palpanas, and Ievgen Redko, *Samformer: Unlocking the potential of transformers in time series forecasting with sharpness-aware minimization and channel-wise attention*, 2024.

[7] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya, *Reformer: The efficient transformer*, 2020.

[8] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan, *Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting*, 2020.

[9] Bryan Lim, Sercan O. Arik, Nicolas Loeff, and Tomas Pfister, *Temporal fusion transformers for interpretable multi-horizon time series forecasting*, 2020.

[10] Jason Lines, Sarah Taylor, and Anthony Bagnall, *Hive-cote: The hierarchical vote collective of transformation-based ensembles for time series classification*, 2016 IEEE 16th International Conference on Data Mining (ICDM), 2016, pp. 1041–1046.

[11] Shizhan Liu, Hang Yu, Cong Liao, Jianguo Li, Weiyao Lin, Alex X. Liu, and Schahram Dustdar, *Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting*, International Conference on Learning Representations, 2022.

[12] Jonathan Long, Evan Shelhamer, and Trevor Darrell, *Fully convolutional networks for semantic segmentation*, 2015.

[13] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam, *A time series is worth 64 words: Long-term forecasting with transformers*, 2023.

[14] Ahmed Shifaz, Charlotte Pelletier, François Petitjean, and Geoffrey I. Webb, *Ts-chief: a scalable and accurate forest algorithm for time series classification*, Data Mining and Knowledge Discovery **34** (2020), no. 3, 742–775.

[15] Zheng Sun, Yi Wei, Wenxiao Jia, and Long Yu, *Stecformer: Spatio-temporal encoding cascaded transformer for multivariate long-term time series forecasting*, 2023.

[16] Sean J. Taylor and Benjamin Letham, *Prophet: Forecasting at scale*, *https://facebook.github.io/prophet/*, 2017, Accedido: 14 de julio de 2024.

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin, *Attention is all you need*, 2023.

[18] Wenxiao Wang, Lu Yao, Long Chen, Binbin Lin, Deng Cai, Xiaofei He, and Wei Liu, *Crossformer: A versatile vision transformer hinging on cross-scale attention*, 2021.

[19] Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun, *Transformers in time series: A survey*, 2023.

[20] Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven Hoi, *Etsformer: Exponential smoothing transformers for time-series forecasting*, 2022.

[21] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long, *Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting*, 2022.

[22] Wang Xue, Tian Zhou, Qingsong Wen, Jinyang Gao, Bolin Ding, and Rong Jin, *Card: Channel aligned robust blend transformer for time series forecasting*, 2024.

[23] Zhenwei Zhang, Linghang Meng, and Yuantao Gu, *Sageformer: Series-aware framework for long-term multivariate time series forecasting*, 2023.

[24] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang, *Informer: Beyond efficient transformer for long sequence time-series forecasting*, 2021.

[25] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin, *Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting*, 2022.