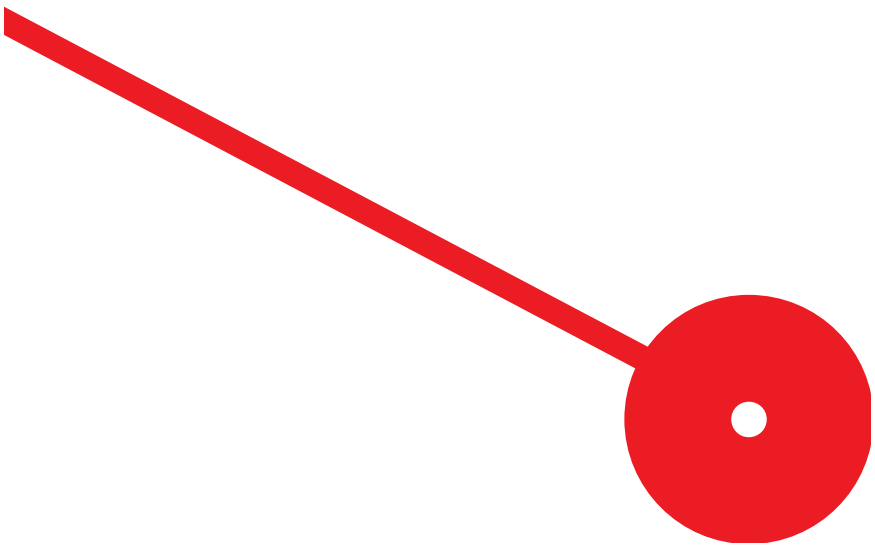
INSTITUTO
SUPERIOR
DE CONTABILIDADE
E ADMINISTRAÇÃO
DO PORTO
POLITÉCNICO
DO PORTO

M MESTRADO
BUSINESS INTELLIGENCE AND ANALYTICS

Transformer-Based Deep Learning Models for Retail Forecasting

Ricardo Jorge Teixeira Caetano

10/2024



Ricardo Jorge Teixeira Caetano. Transformer-Based Deep Learning Models for Retail Forecasting
10/2024

P.PORTO

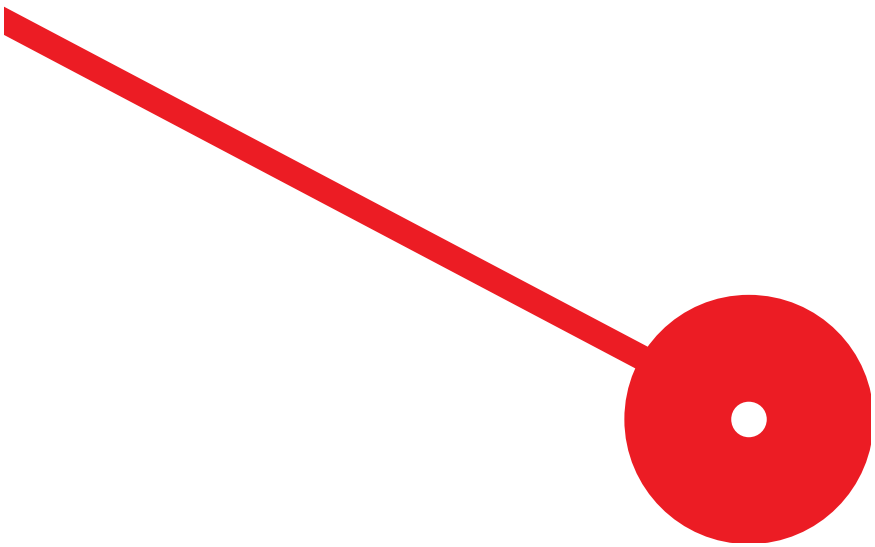
INSTITUTO
SUPERIOR
DE CONTABILIDADE
E ADMINISTRAÇÃO
DO PORTO
POLITÉCNICO
DO PORTO

M MESTRADO
BUSINESS INTELLIGENCE AND ANALYTICS

Transformer-Based Deep Learning Models for Retail Forecasting

Ricardo Jorge Teixeira Caetano

Dissertação de Mestrado apresentada ao Instituto Superior de Contabilidade e Administração do Porto para a obtenção do grau de Mestre em Business Intelligence and Analytics, sob orientação da Doutora Patrícia Alexandra Gregório Ramos e do Doutor José Manuel Soares Oliveira.



Ricardo Jorge Teixeira Caetano. Transformer-Based Deep Learning Models For Retail Forecasting
10/2024

Resumo:

Esta dissertação explora o desempenho de modelos avançados baseados em Transformer para a previsão de séries temporais, com o objetivo de colmatar a lacuna entre os avanços teóricos e as suas aplicações práticas no mundo real. Utilizando o conjunto de dados da competição M5 - que contém mais de 30.000 séries temporais derivadas de dados de vendas em várias categorias de produtos e regiões geográficas - este estudo compara seis modelos Transformer principais: Vanilla Transformer, Autoformer, ETSFormer, Informer, NSTransformer e Reformer. A avaliação foca-se na precisão da previsão, eficiência computacional e robustez, todos fatores críticos para a implementação prática em ambientes reais.

O estudo segue uma abordagem sistemática que envolve uma análise aprofundada das variáveis explicativas, técnicas de pré-processamento e uma afinação extensiva dos hiperparâmetros. Foram utilizadas estratégias avançadas de otimização para identificar as configurações ótimas dos modelos, garantindo um equilíbrio entre o desempenho preditivo e as exigências computacionais. Os modelos foram avaliados com base em métricas tradicionais de previsão pontual, como o *Mean Absolute Scaled Error* e o *Normalized Root Mean Squared Error*, juntamente com métricas probabilísticas como a *Mean Weighted Quantile Loss* e a *Mean Absolute Error Coverage*, de forma a captar tanto a precisão preditiva como a capacidade dos modelos para gerir a incerteza.

Os resultados evidenciam avanços significativos na precisão e robustez dos modelos baseados em Transformer. Os modelos Vanilla Transformer, NSTransformer e ETSformer destacaram-se pela sua capacidade de captar padrões temporais complexos, sem dados adicionais. A análise enfatiza ainda o papel crucial da incorporação de variáveis explicativas e da afinação dos hiperparâmetros para a melhoria dos resultados dos modelos. Foram examinados, os compromissos entre a complexidade do modelo e a eficiência computacional, proporcionando uma perspetiva detalhada sobre a viabilidade prática de implementar estes modelos em contextos operacionais de larga escala.

Em última análise, esta investigação oferece uma visão abrangente dos pontos fortes e dos desafios dos modelos Transformer na previsão de séries temporais.

Palavras-chave: Modelos Transformers, Previsão de Retalho, *Deep Learning*, Análise de series temporais

Abstract:

This research explores the performance of advanced Transformer-based models for time series forecasting, aiming to bridge the gap between theoretical advancements and practical, real-world applications. Using the M5 competition dataset—which contains over 30,000 time series derived from sales data across various product categories and geographic regions—this study benchmarks six key Transformer models: Vanilla Transformer, Autoformer, ETSFormer, Informer, NSTransformer, and Reformer. The evaluation centers around forecasting accuracy, computational efficiency, and robustness, all critical factors for practical deployment.

The study follows a systematic approach that involves a detailed analysis of explanatory variables, preprocessing techniques, and extensive hyperparameter tuning. Advanced optimization strategies were used to identify optimal model configurations, ensuring a balance between forecast performance and computational demands. The models were assessed using traditional point forecasting metrics such as Mean Absolute Scaled Error and Normalized Root Mean Squared Error, alongside probabilistic metrics like Mean Weighted Quantile Loss and Mean Absolute Error Coverage, to capture both predictive accuracy and the models' ability to manage uncertainty.

The findings underscore notable advancements in accuracy and robustness for Transformer-based models, particularly Vanilla Transformer, NSTransformer and ETSformer, which stood out in their ability to capture intricate temporal patterns, without features. The analysis also emphasizes the critical role of incorporating explanatory variables and tailoring hyperparameter settings to improve model outcomes. Trade-offs between model complexity and computational efficiency are carefully examined, providing a nuanced perspective on the practical feasibility of deploying these models in large-scale operational contexts.

Ultimately, this study offers a comprehensive view of the strengths and challenges of Transformer models in time series forecasting, providing practical guidance for their effective implementation in dynamic and complex environments that are marked by high variability and uncertainty.

Keywords: Transformer Models, Retail Forecasting, Deep Learning, Time Series Analysis

List of Abbreviations

AI – Artificial Intelligence

ARIMA – Autoregressive Integrated Moving Average

CNN – Convolutional Neural Networks

CRPS – Continuous Ranked Probability Score

DGP – Deep Gaussian Processes

ESA – Exponential Smoothing Attention

ESN – Echo State Networks

ETS – Exponential Smoothing State Space Models

FA – Frequency Attention

FEDformer – Frequency Enhanced Decomposition Transformer

FFT – Fast Fourier Transform

GAN – Generative Adversarial Networks

GNN – Graph Neural Networks

GRU – Gated Recurrent Units

LSTM – Long Short-Term Memory

LSTF – Long Sequence Time-Series Forecasting

MAE – Mean Absolute Error

MAEC –Mean Absolute Error Coverage

MASE – Mean Absolute Scaled Error

MDN – Mixture Density Network

MLP – Multilayer Perceptrons

MSE – Mean Squared Error

MWQL – Mean Weighted Quantile Loss

NLL – Negative Log-Likelihood

NN – Neural Networks

NRMSE – Normalized Root Mean Squared Error

RMSE – Root Mean Squared Error

RNN – Recurrent Neural Networks

TCN – Temporal Convolutional Networks

TFT – Temporal Fusion Transformer

WQL – Weighted Quantile Loss

Table of Contents

List of Figures

List of Tables

1 Introduction

Forecasting time series data has become a cornerstone in many industries, including finance, healthcare, retail, and energy, where understanding future trends is essential for making strategic decisions. Accurate time series forecasting allows organizations to manage supply chains, optimize inventory, allocate resources effectively, and mitigate risks. Over recent years, advances in machine learning—particularly deep learning—have brought remarkable new capabilities to this field. This thesis explores the use of Transformer-based models, originally developed for natural language processing, to improve forecasting accuracy and efficiency, especially in complex and hierarchical retail data.

Traditionally, time series forecasting was the domain of statistical models like Autoregressive Integrated Moving Average (ARIMA) and Exponential Smoothing State Space Models (ETS). While these models work reasonably well for stable environments with linear and stationary data, they often fall short in dealing with modern, volatile datasets that exhibit non-linear patterns, temporal dynamics, and hierarchical structures (Hyndman & Athanasopoulos, 2021). The emergence of Recurrent Neural Networks (RNN), including variants like Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU), offered a promising approach to modeling temporal relationships. However, these architectures often struggle with capturing long-term dependencies and suffer from inefficiencies such as the vanishing gradient problem (Goodfellow et al., 2016).

The breakthrough came with the introduction of Transformer architectures, which leverage self-attention mechanisms to understand relationships within a sequence of data points, regardless of their relative distance. Initially developed for natural language processing by Vaswani et al., (2017), Transformers have demonstrated unique advantages in time series applications, such as parallelization of data processing, the ability to model complex temporal dependencies, and scalability to large datasets (Wu et al., 2021). These features make them well-suited to forecasting tasks in domains with intricate time dynamics, including retail. Unlike sequential RNN models, Transformers capture temporal patterns more effectively by analyzing the entire sequence in parallel, thereby avoiding the pitfalls of traditional sequential models.

In the retail domain, accurate forecasting directly influences key operational and strategic decisions, such as workforce planning, inventory management, promotional strategies, and revenue optimization. The M5 competition dataset (Howard et al., 2020), hosted by Kaggle in collaboration with Walmart and the University of Nicosia, has emerged as a benchmark for evaluating forecasting models in retail, given its scale, diversity, and the hierarchical nature of the data (Makridakis et al., 2022). The dataset includes daily sales data of over 3,000 products across ten stores in the United States, reflecting the challenges faced in real-world retail—fluctuating demand, varied seasonal patterns, and the effect of promotional events. The complexity and hierarchical structure of this dataset make it an ideal candidate for evaluating the effectiveness of Transformer-based models, particularly in predicting both granular, product-level demand and aggregate trends across categories.

This thesis aims to assess the applicability and performance of several Transformer-based models for forecasting tasks using the M5 dataset as benchmark, with a particular focus on their ability to provide accurate point and probabilistic forecasts. Six models are being evaluated: the Vanilla Transformer, Autoformer, ETSFormer, Informer, NSTransformer, and Reformer. Each of these models extends the core Transformer architecture in unique ways. Informer, for instance, integrates a sparse attention mechanism to improve efficiency with long sequences, whereas Autoformer introduces decomposition methods to separate trend and seasonal components of the data (H. Zhou et al., 2021). The inclusion of these variations allows a nuanced exploration of how different architectural enhancements contribute to forecasting accuracy and efficiency.

Hyperparameter tuning is another critical aspect of this study. Deep learning models, including Transformers, are highly sensitive to their configuration, such as the number of attention heads, depth of encoder layers, dropout rates, and learning rates. In this research, systematic hyperparameter optimization is carried out using the Optuna framework to identify the optimal configuration for each model (Bengio, 2012). The focus is not only on improving model accuracy but also on ensuring computational efficiency, a crucial consideration given the scalability requirements in retail forecasting, where models must be retrained frequently and deployed at scale.

Another key contribution of this thesis is the use of explanatory variables to enhance model performance. These variables, which include temporal information, pricing details, promotional events, and product-level identifiers, provide context that significantly

impacts demand patterns (Salinas et al., 2019). Including such features can help Transformer models better understand and predict sales behavior. For instance, knowing the day of the week or whether there is an upcoming holiday allows models to anticipate surges or dips in sales, thus improving the robustness of both point and probabilistic forecasts.

This thesis evaluates the models based on both point forecasting and probabilistic forecasting metrics. Point forecasting provides a single expected future value, while probabilistic forecasting offers a distribution of possible outcomes. This dual approach is particularly relevant in retail, where uncertainty is a constant factor and over- or under-forecasting can have significant implications for inventory costs, stockouts, or wastage. Metrics such as Normalized Root Mean Squared Error (NRMSE), Mean Absolute Scaled Error (MASE), Mean Weighted Quantile Loss (MWQL) and Mean Absolute Error (MAE) Coverage are used to evaluate both the accuracy of point estimates and the reliability of probabilistic forecasts. Each of these metrics provides different insights. For instance, NRMSE penalizes large errors more severely, making it suitable for understanding significant prediction deviations, while MWQL helps gauge the accuracy across different quantiles, which is crucial for inventory optimization in uncertain environments (Taylor & Letham, 2018).

The contributions of this study are threefold. First, it provides a comprehensive benchmarking of six Transformer-based architectures for time series forecasting in the context of retail. The comparisons made across different metrics help establish a clearer picture of which model features contribute to forecasting effectiveness in the presence of hierarchical and complex temporal dynamics. Second, it highlights the role of hyperparameter tuning and the use of explanatory variables in enhancing forecasting performance. Careful configuration and context inclusion are shown to significantly improve model robustness and accuracy. Lastly, by examining both point and probabilistic forecasts, this research offers valuable insights into the practical applicability of Transformer-based models for decision-making in volatile retail environments.

The structure of this thesis is organized as follows: Chapter II discusses the state-of-the-art in time series forecasting, focusing on the evolution from traditional statistical models to modern deep learning approaches. Chapter III details the methodology, including the models selected, data preprocessing steps, and the hyperparameter

optimization process. Chapter IV presents the empirical study, including dataset descriptions, evaluation metrics, and an in-depth analysis of the results. Finally, Chapter V concludes with a summary of findings, practical implications, and suggestions for future research directions in time series forecasting.

This research aims to bridge the gap between theoretical advancements in machine learning and practical applications in retail forecasting, leveraging Transformer architectures to enhance predictive accuracy while addressing scalability challenges. In doing so, it contributes to the growing body of literature that underscores the potential of advanced machine learning techniques to revolutionize decision-making processes in complex, data-driven environments.

2 State of the Art

The field of time series forecasting has evolved significantly over recent years, driven by advances in neural network architectures and deep learning techniques. The application of deep learning to time series forecasting began with the use of Recurrent Neural Networks (RNN). RNN, including Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRU), were designed to handle sequential data by maintaining an internal state that evolves over time (Hochreiter & Schmidhuber, 1997). These models are well-suited for capturing temporal dependencies, making them popular for early applications in time series forecasting. However, RNN face several limitations, such as the vanishing gradient problem, which hinders their ability to model long-term dependencies effectively (Gal & Ghahramani, 2016).

To overcome these limitations, Convolutional Neural Networks (CNN) and their variant, Temporal Convolutional Networks (TCN), were introduced. TCN use dilated convolutions to capture long-range dependencies without the need for recurrent connections, offering advantages in parallelization and stable training (Bai et al., 2018). While TCN improve training efficiency and stability, they still face challenges in modelling complex, non-linear relationships over long sequences, which are common in real-world datasets like the M5 – Forecasting Accuracy (Howard et al., 2020).

The introduction of Transformer models marked a significant breakthrough in time series forecasting. Transformers utilize self-attention mechanisms to capture dependencies between all-time steps in a sequence simultaneously (Vaswani et al., 2017). This capability is particularly useful for long-term forecasting and handling irregular patterns, making Transformers and their variants highly effective for complex time series datasets.

These advancements have introduced sophisticated tools capable of learning complex temporal relationships, thus improving accuracy and robustness across a variety of applications. Neural network models, such as Multilayer Perceptrons (MLP), Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Transformer-based models, have transformed how time series data is analysed and predicted.

Each of these models addresses different aspects of the challenges inherent in time series forecasting, from capturing local temporal dependencies to handling long-term

sequential information. Moreover, recent innovations in deep learning, including probabilistic forecasting and Transformer variants, have further pushed the boundaries of what is possible in this domain, offering better handling of uncertainty, scalability, and interpretability.

This section provides a comprehensive overview of the state-of-the-art techniques employed in time series forecasting, focusing on the evolution and impact of neural network architectures and deep learning approaches. The aim is to highlight the progression from traditional methods to the latest sophisticated models, elucidating their strengths, challenges, and the opportunities they present for future improvements in time series forecasting.

2.1 Neural Network Architectures

Neural network architectures have significantly advanced the field of time series forecasting, providing highly flexible and powerful tools for capturing nonlinear relationships in data. In the past few decades, machine learning has undergone a rapid evolution, with Neural Networks (NN) playing a central role in this transformation. Initially developed for simpler classification and regression tasks, NN have evolved to tackle complex problems involving sequential and time-dependent data. The capacity of NN to learn from raw data and model intricate relationships has made them an attractive tool for time series forecasting, an area that deals with predicting future values based on previously observed data points.

The primary advantage of NN lies in their ability to approximate almost any function given sufficient data and model complexity. This makes them highly versatile for capturing the non-linear, temporal dependencies that are often present in real-world time series datasets. As time series forecasting applications span numerous fields including finance, energy, healthcare, and supply chain management, there has been a notable shift towards leveraging deep learning techniques to improve forecasting accuracy and robustness.

Recent years have witnessed significant advancements in the architecture of neural networks for time series forecasting. From early approaches using basic Multilayer Perceptrons to more sophisticated architectures like Convolutional Neural Networks, Recurrent Neural Networks, and Transformer models, each new architecture has brought unique strengths in handling different aspects of time series data. These developments

have allowed for improved scalability, better handling of complex temporal relationships, and enhanced interpretability. This section explores the state-of-the-art neural network architectures, detailing their applications, strengths, and challenges in the context of time series forecasting.

2.1.1 Multilayer Perceptron

Multilayer Perceptrons (MLP), or feedforward neural networks, are among the simplest forms of neural networks, characterized by layers of interconnected nodes (neurons). MLP consist of an input layer, one or more hidden layers, and an output layer. Each hidden layer applies an affine transformation followed by a nonlinear activation, enabling the network to approximate complex functions. The fully connected nature of MLP makes them suitable for capturing generalized nonlinear relationships, but this also makes them computationally expensive when dealing with high-dimensional input data, such as time series (Benidis et al., 2020).

One of the main limitations of MLP for time series forecasting is their inability to directly exploit the sequential structure inherent in temporal data. This lack of temporal sensitivity necessitates the use of more advanced neural architectures like Convolutional Neural Networks and Recurrent Neural Networks, which are better suited for sequential data (Benidis et al., 2020; Hochreiter & Schmidhuber, 1997).

2.1.2 Convolutional Neural Networks

Convolutional Neural Networks (CNN) were initially designed for computer vision tasks but have found applications in time series forecasting due to their ability to extract localized features from sequential data. In the context of time series, CNN utilize convolutional layers to capture temporal patterns by applying a kernel (or filter) that slides across the input. This allows the model to detect important features in local temporal windows, making CNN highly effective for tasks where short-term dependencies are crucial (Benidis et al., 2020; Lecun et al., 1998).

A specific variation known as causal convolution ensures that predictions at a given time step only depend on past values, preserving the causality in time series forecasting. Dilated causal convolutions further extend this by expanding the receptive field, allowing the model to consider longer sequences without a proportional increase in computational cost (Benidis et al., 2020).

2.1.3 Recurrent Neural Networks and Long Short-Term Memory

Recurrent Neural Networks are designed to handle sequential data by incorporating cycles within their architecture, which allows the network to maintain a form of memory through hidden states that are shared across time steps. This makes RNN well-suited for modeling time-dependent data. However, training traditional RNN can be challenging due to issues such as vanishing or exploding gradients, particularly when dealing with long sequences (Benidis et al., 2020).

To address these issues, more advanced variations like Long Short-Term Memory networks and Gated Recurrent Units have been introduced. LSTM add specialized gating mechanisms that regulate the flow of information, making it possible for the model to retain long-term dependencies while mitigating the issues of gradient vanishing. This capability is especially useful for complex forecasting tasks where the relationships span extended time horizons (Benidis et al., 2020; Hochreiter & Schmidhuber, 1997).

2.1.4 Transformer Models

The Transformer architecture, originally developed for natural language processing tasks (Vaswani et al., 2017), has recently gained traction in time series forecasting due to its attention mechanism, which allows the model to focus on specific parts of the input sequence that are most relevant for making predictions. Unlike RNN, Transformers do not rely on sequential processing, which makes them highly efficient and capable of capturing long-range dependencies in a more scalable manner (Benidis et al., 2020; Vaswani et al., 2017).

A notable advantage of Transformer models in time series forecasting is their ability to parallelize computations, making them suitable for large-scale industrial applications where forecasting thousands of time series simultaneously is required. The success of the Transformer architecture in forecasting applications is also attributed to its flexibility in incorporating various covariates and handling irregularly spaced time series (Benidis et al., 2020).

The development of neural network architectures for time series forecasting has moved from simpler MLP to more sophisticated and specialized architectures like CNN, RNN, LSTM, and Transformers. Each of these architectures brings unique advantages:

- **MLP** are straightforward and effective for generalized tasks but lack temporal awareness.
- **CNN** are capable of capturing local temporal patterns efficiently.
- **RNN** and **LSTM** excel at modeling long-term dependencies in sequential data.
- **Transformers** leverage attention mechanisms to handle long-range dependencies and are well-suited for parallel processing.

These diverse architectures reflect the growing trend in leveraging deep learning to push the boundaries of time series forecasting accuracy, enabling better scalability, robustness, and interpretability across a wide range of applications (Benidis et al., 2020).

2.2 Deep Learning for Time Series Forecasting

Deep learning has emerged as one of the most powerful tools for addressing the challenges of time series forecasting, especially due to its ability to model complex, nonlinear temporal dependencies. In this section we will review the key deep learning models used for short-term time series forecasting, focusing on recent advances in Convolutional Neural Networks, Temporal Convolutional Networks, Recurrent Neural Networks, Graph Neural Networks (GNN), Deep Gaussian Processes (DGP), Generative Adversarial Networks (GAN), and Diffusion Models, as outlined by (Casolaro et al., 2023).

2.2.1 Convolutional Neural Networks and Temporal Convolutional Networks

CNNs have shown their potential in capturing localized temporal patterns through convolutional layers that slide over time series data. Temporal Convolutional Networks build upon CNN by introducing causal and dilated convolutions, ensuring that predictions at a specific time step are informed only by past values, thereby maintaining the temporal causality of the sequence (Casolaro et al., 2023). TCN also enable an increased receptive field, allowing for better modeling of longer-term dependencies compared to conventional CNN.

2.2.2 Recurrent Neural Networks and Variants

RNNs are specifically designed to handle sequential data by maintaining a form of memory through their recurrent connections. Variants of RNN such as Elman RNN, Echo State Networks (ESN), Long Short-Term Memory networks, and Gated Recurrent Units have been developed to address the limitations of standard RNN, such as unstable

training and difficulty in modeling long-term dependencies (Casolaro et al., 2023). LSTM and GRU, in particular, introduce gating mechanisms that regulate the flow of information, allowing the network to better capture long-term dependencies while mitigating the vanishing gradient problem.

2.2.3 Graph Neural Networks and Deep Gaussian Processes

Graph Neural Networks have recently gained attention for multivariate time series forecasting, particularly for scenarios where spatial dependencies are present. GNN can model the relationships among different time series by representing them as nodes in a graph, enabling the extraction of useful features from these interdependencies (Casolaro et al., 2023).

Deep Gaussian Processes extend traditional Gaussian Processes by stacking multiple layers, enabling them to capture more complex, hierarchical relationships in the data. Unlike other deep learning models, DGP not only provide point estimates for predictions but also quantify uncertainty, which is crucial for decision-making processes in sensitive applications (Casolaro et al., 2023).

2.2.4 Generative Adversarial Networks and Diffusion Models

Generative Adversarial Networks have been employed in time series forecasting both as data augmentation tools and as end-to-end forecasting models. GAN generate synthetic data to enhance the training dataset, particularly when only limited real data is available. Diffusion Models, on the other hand, are a more recent innovation that progressively injects noise into data and learns to reverse the process to make predictions, providing a new way of tackling short-term forecasting tasks with a stochastic framework (Casolaro et al., 2023).

2.3 Transformers for Time Series Forecasting

Transformers represent one of the most influential advancements in deep learning for time series forecasting, especially in the context of long-term forecasting. Originally designed for natural language processing, the Transformer architecture has been adapted to time series forecasting, taking advantage of its powerful attention mechanisms to model long-range dependencies without the limitations of traditional RNN (Vaswani et al., 2017).

Transformers are well-suited for long-term forecasting because of their ability to handle sequential data in a parallelized manner. This is achieved through the self-attention mechanism, which allows the model to selectively focus on relevant parts of the input sequence regardless of their distance from the target. This overcomes the short-term memory problem faced by RNN and LSTM, which struggle to maintain information over long sequences.

2.3.1 Self-Attention Mechanism and Long-Term Dependencies

The self-attention mechanism is central to the Transformer's power in modeling time series. Unlike RNN, which process inputs sequentially, Transformers compute attention scores for every input element relative to every other element, thereby establishing connections across the entire sequence. This property is particularly useful for long-term time series forecasting where the model needs to capture dependencies that span across multiple time points (Vaswani et al., 2017).

In long-term forecasting scenarios, such as energy load prediction or financial market trends, understanding these long-range dependencies is crucial. Transformer-based models, such as Informer and Autoformer, have been specifically developed to improve the efficiency and scalability of Transformers for time series tasks (Wu et al., 2021; H. Zhou et al., 2021).

2.3.2 Transformer Variants for Long-Term Forecasting

Several Transformer variants have been proposed to address the limitations of the original architecture when applied to time series forecasting. These include Informer, Autoformer, and FEDformer, each introducing novel mechanisms to enhance efficiency and accuracy in long-term forecasting.

The Informer model addresses the challenge of handling long sequences in time series data by introducing a probabilistic sparse self-attention mechanism (H. Zhou et al., 2021). This mechanism reduces the computational burden by focusing only on the most informative components of the sequence, rather than computing attention for every possible pair. Informer also uses distilling operations to compress sequences, allowing the model to handle longer sequences more effectively without a proportional increase in computational cost.

The Autoformer model introduces auto-correlation mechanisms to better capture seasonal patterns and long-term trends in time series forecasting (Wu et al., 2021). This mechanism helps the model to automatically aggregate and extract key features from long sequences, which is particularly useful for tasks involving data with inherent periodicity, such as temperature or energy consumption prediction. The Autoformer's design is also intended to avoid the performance degradation seen in standard Transformers when dealing with longer input sequences.

FEDformer (Frequency Enhanced Decomposition Transformer) takes a novel approach by decomposing time series into components across different frequency bands. This allows the model to selectively focus on patterns relevant to different time scales, improving its ability to forecast long-term dependencies and capture both low- and high-frequency variations effectively. This frequency decomposition is particularly advantageous in time series that contain multiple seasonal patterns or varying frequencies over time (T. Zhou et al., 2022).

2.3.3 Challenges and Opportunities

While Transformers have shown promise in time series forecasting, there are several challenges that need to be addressed for optimal performance:

- **Memory Bottleneck**: The original Transformer has quadratic complexity concerning the sequence length due to the self-attention mechanism. This makes it computationally expensive, particularly for long-term forecasting tasks with large datasets (H. Zhou et al., 2021).
- **Locally Agnostic Nature**: The Transformer's attention mechanism is inherently global, which can overlook important local temporal patterns that are crucial for accurate forecasting in certain domains (Wu et al., 2021).
- **Stationarization Issues**: The Transformer architecture may struggle with non-stationary time series, where statistical properties such as the mean and variance change over time. This can lead to poor model performance when trained on time series data without proper handling of non-stationarity (Liu et al., 2022).

To overcome these limitations, several variants of the Transformer architecture have been proposed, each introducing modifications to improve efficiency and accuracy in time series forecasting:

- **LogTrans**: Li et al., (2019) introduced LogTrans, a Transformer variant that addresses both the memory bottleneck and the locally agnostic nature of the standard Transformer. By incorporating causal convolutions and a log-sparse attention mechanism, LogTrans reduces the computational burden while preserving important local structures in the data.
- **Informer**: H. Zhou et al., (2021) proposed the Informer model, which introduces a sparse attention mechanism to focus only on the most informative queries, reducing the complexity of the self-attention operation from quadratic to logarithmic. Informer also features a self-attention distillation mechanism that aggregates information from long input sequences more effectively.
- **Autoformer**: Wu et al., (2021) designed Autoformer to improve the Transformer's ability to model time series with periodic patterns. The Autoformer replaces the standard attention mechanism with an autocorrelation-based approach that captures long-term dependencies through time-delay similarity. Additionally, it decomposes time series into trend and seasonal components, allowing the model to handle non-stationarity more effectively.
- **FEDFormer**: T. Zhou et al., (2022) developed the FEDFormer, which reduces the complexity of attention mechanisms by applying them in the frequency domain. By representing time series data using Fourier or Wavelet transforms, FEDFormer is able to capture long-term dependencies with lower computational costs.
- **Crossformer**: Zhang & Yan, (2022) introduced Crossformer, which processes each dimension of multivariate time series independently using a cross-attention mechanism to model interdependencies between variables. This model segments input sequences into non-overlapping sub-sequences and applies attention mechanisms at multiple scales, improving its ability to capture both short- and long-term dependencies.
- **PatchTST**: Nie et al., (2023) proposed PatchTST, which segments time series data into smaller patches and applies attention mechanisms within each patch. This approach enables the model to capture local temporal patterns more effectively, while still leveraging the global attention mechanism for long-term dependencies.

- **Pyraformer**: Liu et al., (2022) proposed the Pyraformer, which utilizes a pyramidal attention mechanism that progressively aggregates information across different time scales. This model is designed to capture both short-term and long-term dependencies in a computationally efficient manner, making it suitable for long-range forecasting tasks.

The performance of these Transformer variants has been benchmarked on a variety of time series forecasting datasets, including retail, electricity, and stock market data. Metrics such as Mean Squared Error (MSE) and Mean Absolute Error (MAE) are commonly used to evaluate model performance across different prediction lengths (Wu et al., 2021; H. Zhou et al., 2021).

While each variant offers improvements over the original Transformer, there is ongoing research to further reduce computational complexity, improve handling of non-stationarity, and enhance interpretability in time series forecasting models.

Transformer-based models have become state-of-the-art for time series forecasting, particularly for long-term predictions where capturing dependencies across time is crucial. Continued advancements in attention mechanisms, efficiency improvements, and handling of non-stationary data are likely to drive further improvements in this area.

2.4 Probabilistic Time Series Forecasting

Probabilistic time series forecasting aims to predict the full range of possible future values rather than providing a single point estimate. This approach is crucial for decision-making in many applications where understanding uncertainty and risk is necessary, such as finance, supply chain management, and energy demand forecasting. Unlike traditional deterministic models, probabilistic forecasting captures the inherent uncertainty in time series data, providing not just a forecasted value but also a confidence interval or distribution.

2.4.1 Output Models and Loss Functions

The probabilistic approach to time series forecasting relies heavily on choosing appropriate output models and loss functions, which influence how uncertainty is captured and propagated. According to Benidis et al., (2020), output models for probabilistic forecasting include distributional assumptions like Gaussian or Laplace

distributions. The model generates probabilistic forecasts by estimating the parameters of these distributions, quantifying the uncertainty of predictions.

Loss functions for probabilistic forecasting differ from those used in traditional point forecasting. For example, the Negative Log-Likelihood (NLL) is often used instead of Mean Squared Error because it focuses on optimizing the likelihood of observing the true outcomes, given the predicted distribution. This helps the model better capture the full distribution rather than just a mean estimate.

Quantile regression is also widely used in probabilistic forecasting to predict different quantiles of the output distribution, thereby providing multiple levels of uncertainty. Another technique employed is the Mixture Density Network (MDN), which models the output as a mixture of multiple probability distributions. This allows the model to handle complex, multimodal relationships in time series data effectively (Benidis et al., 2020).

2.4.2 Deep Learning Approaches

Deep learning models have incorporated probabilistic elements to improve forecasting. A notable example is DeepAR, developed by Amazon, which is an RNN-based approach that estimates the parameters of a predefined probability distribution such as Gaussian. This allows the model to provide probabilistic forecasts by sampling from the estimated distribution. Another example is the Temporal Fusion Transformer (TFT), which produces probabilistic outputs that quantify uncertainty, enhancing decision support in critical applications (Benidis et al., 2020).

2.4.3 Advantages of Probabilistic Forecasting

Probabilistic forecasting provides richer insights compared to point forecasting by offering a range of possible outcomes rather than just the most likely future value. This is especially important for risk management and decision-making in fields like finance, where understanding the distribution of potential outcomes is crucial. For example, forecasting future electricity demands not only requires a point estimate but also the confidence intervals to ensure robust planning and prevent overestimation or underestimation, which can be costly.

To evaluate the quality of probabilistic forecasts, metrics such as the Continuous Ranked Probability Score (CRPS) are employed. CRPS assesses the accuracy of

probabilistic forecasts by comparing the predicted cumulative distribution to the observed outcome, providing a more comprehensive evaluation than metrics like MAE or RMSE.

Overall, probabilistic time series forecasting is an essential evolution in the field, providing a more complete picture of future possibilities by capturing and modeling uncertainty. By leveraging advanced loss functions and sophisticated output models, modern deep learning-based probabilistic forecasting approaches such as DeepAR and TFT have moved beyond simple point estimates. They offer valuable tools for risk assessment and decision support across various sectors (Benidis et al., 2020).

3 Methodology

This chapter outlines the methodological framework employed in this research, specifically focusing on advancements in Transformer-based architectures for time-series forecasting. Time-series data, characterized by temporal dependencies and evolving patterns, presents unique challenges, particularly for long-term forecasting in diverse domains such as finance, energy, and healthcare. While traditional neural network architectures like Recurrent Neural Networks and Long Short-Term Memory networks have demonstrated effectiveness in modeling sequential dependencies, they are limited by sequential processing constraints, which restrict their ability to capture long-range dependencies efficiently. The advent of the Transformer model by Vaswani et al., (2017) marked a significant shift by introducing a self-attention mechanism that enables parallel processing, thereby improving the model's capacity to capture complex patterns over extended sequences.

This research applies and compares several Transformer-based architectures tailored for time-series data, including the Vanilla Transformer, Autoformer, ETSformer, Informer, NSTransformer, and Reformer models. Each of these models builds upon the original Transformer architecture to address specific challenges in time-series forecasting, such as handling non-stationarity, reducing computational complexity, and improving the model's interpretability. This chapter delves into the architectural modifications and innovative mechanisms within these models, such as decomposition frameworks, novel attention mechanisms, and adaptations for non-stationary data, to provide a comprehensive understanding of their methodological approaches.

The methodological choices in this research are guided by the need to balance computational efficiency with model accuracy and interpretability. Each model’s architecture, from the core self-attention mechanism in the Vanilla Transformer to the decomposition-based framework in Autoformer and the Exponential Smoothing Attention in ETSformer, is examined to highlight how these innovations contribute to capturing temporal dependencies effectively. Through a structured approach to model comparison and evaluation, this chapter seeks to establish a clear framework for analyzing the efficacy of Transformer-based models in addressing the unique demands of time-series forecasting.

3.1 Vanilla Transformer

The Vanilla Transformer, first introduced by Vaswani et al., (2017) has had a significantly impact on the field of neural networks, particularly for sequential data tasks such as natural language processing (NLP). Unlike traditional models like Recurrent Neural Networks and Long Short-Term Memory Networks, which process sequences sequentially, the Transformer relies entirely on self-attention mechanisms, enabling it to capture long-range dependencies more efficiently. This section outlines the core architecture and components of the Vanilla Transformer, emphasizing its relevance to time series forecasting.

The model is structured around an encoder-decoder framework, as seen in Figure 1, where each component comprises multiple layers with identical sub-layer configurations. Each encoder and decoder layer incorporates two key sub-layers: a multi-head self-attention mechanism and a position-wise fully connected feed-forward network. Both sub-layers are augmented with layer normalization and residual connections, enhancing training stability and facilitating deeper architectures.

The encoder is composed of N identical layers, each with the following components:

1. **Multi-Head Self-Attention Mechanism:** allows each position in the input sequence to attend to all other positions, generating context-aware representations. It calculates three matrices for each input: queries Q, keys K, and values V. The attention weights are derived from the scaled dot-product of Q and K, defined as:

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V,$$

   where d_k is the dimension of the key vectors. These attention scores are used to compute a weighted sum of the values V, effectively capturing dependencies between different positions.

2. **Feed-Forward Network,** following the self-attention layer, a position-wise feed-forward network is applied to each position separately and identically. It consists of two linear transformations with a $ReLU$ activation function in between, as expressed by:

$$\mathrm{FFN}(x) = \max\left(0, xW_1 + b_1\right) W_2 + b_2.$$

This sub-layer allows the model to learn complex representations by introducing non-linearity.

The decoder mirrors the encoder's structure, with an additional layer that incorporates multi-head attention over the encoder's outputs. This facilitates effective integration of the encoded input information during the generation of output sequences.

- Masked Multi-Head Self-Attention:

The decoder utilizes a similar self-attention mechanism as the encoder but with masking to prevent the model from attending to future positions during training. This ensures that the prediction for a position t only depends on positions $< t$.

- Encoder-Decoder Attention:

This additional layer allows the decoder to focus on specific parts of the input sequence by attending to the encoder outputs. The computed attention weights determine the relevance of different input tokens, effectively linking input and output sequences.

- Feed-Forward Network:

Similar to the encoder, the decoder applies a feed-forward network to each position, enabling richer representations through non-linear transformations.

Since the Transformer model lacks inherent sequential information due to its parallel structure, positional encodings are added to the input embeddings to incorporate the sequence order. These encodings are defined using sinusoidal functions and are calculated as follows:

$$\mathrm{PE}(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/\,dmodel}}\right),$$

$$\mathrm{PE}(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/\,dmodel}}\right),$$

where pos represents the position, i denotes the dimension, and d_{model} is the model's dimensionality. This addition of positional encodings allows the model to effectively capture the order and relative distance of elements within the sequence.

The self-attention mechanism is fundamental to the Transformer's ability to model dependencies irrespective of their distance in the sequence. It projects the input embeddings into queries, keys, and values, and computes the attention scores as described in the equation above. The multi-head attention mechanism extends this by computing several sets of attention scores in parallel, each with its unique learnable projections. This is particularly beneficial for capturing different types of dependencies, as shown in Figure 1.

Mathematically, the multi-head attention is defined as:

$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1,\text{head}_2,...,\text{head}_h)W^O,$$

where $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$ and W_i^Q, W_i^K, W_i^V are learned projections for each head. This mechanism enables the model to attend to different positions based on multiple aspects of the input sequence.

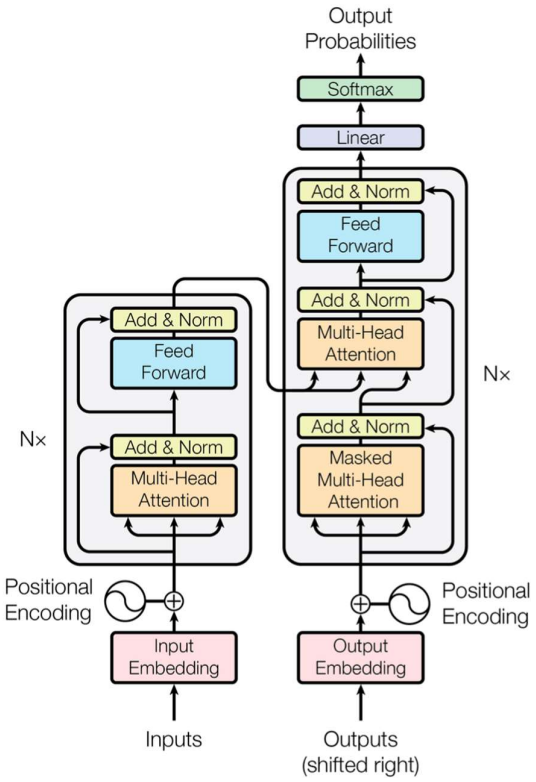


Figure 1 - Transformer model architecture. Reprinted from (Vaswani et al., 2017).

3.2 Autoformer

Autoformer, introduced by Wu et al., (2021), represents a significant advancement in the domain of time series forecasting, particularly addressing the complexities associated with long-term predictions in real-world scenarios. Traditional Transformer models, despite their effectiveness in capturing dependencies within sequential data, face substantial challenges when applied to long-term forecasting. These challenges stem from their quadratic computational complexity and memory usage, as well as their limited ability to model complex temporal patterns over extended periods. To overcome these limitations, Autoformer proposes a novel architecture that integrates a decomposition-based framework with an innovative Auto-Correlation mechanism, fundamentally rethinking the role of Transformers in time series forecasting.

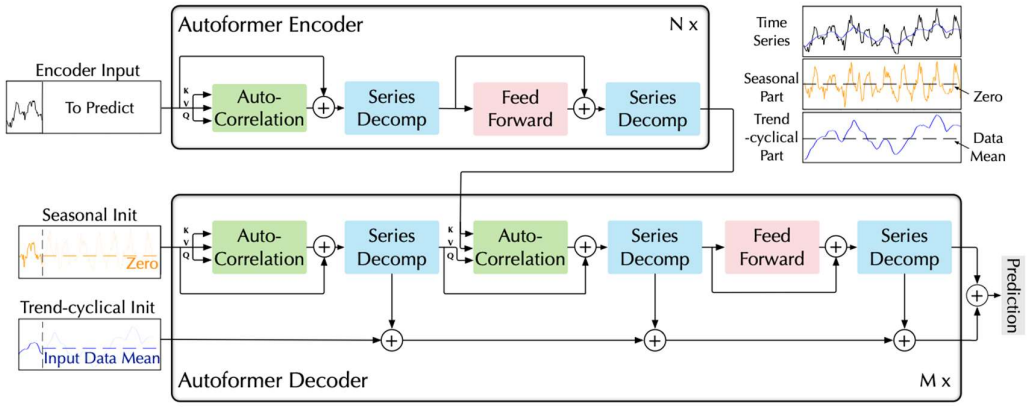


Figure 2 - Autoformer - The encoder-decoder structure with integrated decomposition blocks and the Autocorrelation mechanism. Reprinted from (Wu et al., 2021).

The model's decomposition architecture, as seen in the Figure 2, embedded within both the encoder and decoder, allows for the continuous refinement of temporal representations. It iteratively separates the input series into trend-cyclical and seasonal components using a series of internal moving average operations. This progressive decomposition enables Autoformer to effectively capture both short-term seasonality and long-term trends, enhancing forecasting accuracy.

To overcome the inefficiencies of traditional self-attention mechanisms, Wu et al., (2021) introduces the Auto-Correlation mechanism, as seen in Figure 3. This mechanism leverages the periodic nature of many time series, identifying dependencies at the sub-series level and using Fast Fourier Transform (FFT) for efficient computation. This

reduces computational complexity to $O(L \log L)$ and allows the model to capture long-range dependencies more effectively.

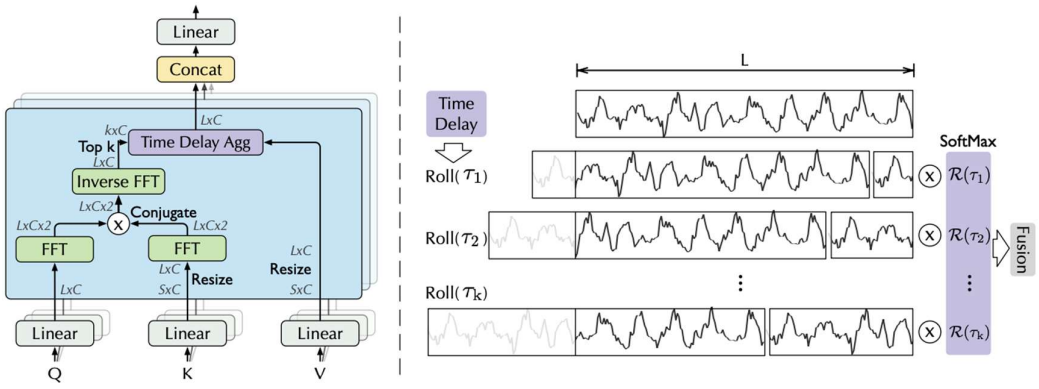


Figure 3 - Autocorrelation (left) and Time Delay Aggregation (right). Reprinted from (Wu et al., 2021).

Time Delay Aggregation, as shown in Figure 3, refers to the process of aligning and aggregating sub-series in the time series data based on their identified periodic patterns. By shifting the sub-series according to selected time delays, the model aligns similar patterns occurring at the same phase positions across different periods. These aligned sub-series are then aggregated using a weighted sum derived from their autocorrelation values, effectively capturing repetitive temporal dependencies and providing a more accurate representation of long-range patterns within the data.

3.3 ETSformer

The ETSformer, introduced by Woo et al., (2022), represents a significant advancement in the application of Transformers to time-series forecasting. Traditional Transformers, although powerful, are not inherently designed to capture the unique characteristics of time-series data, such as seasonality, trends, and hierarchical dependencies. ETSformer addresses these challenges by integrating principles of exponential smoothing into a Transformer-based architecture, effectively modelling level, growth, and seasonal components of time-series data (Woo et al., 2022)

As illustrated in Figure 4, the ETSformer architecture follows an encoder-decoder structure tailored to decompose time-series data into interpretable components: level, growth, and seasonality. Each component is extracted iteratively through multiple layers in both the encoder and decoder, leveraging novel mechanisms like Exponential

Smoothing Attention (ESA) and Frequency Attention (FA) to enhance the model's ability to capture temporal dependencies and periodic patterns.

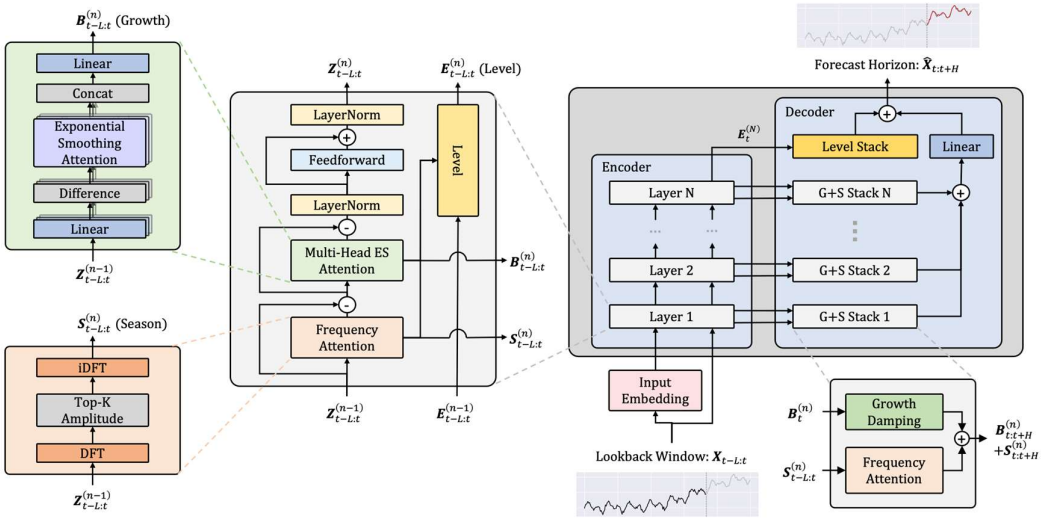


Figure 4 - ETSformer model architecture. Reprinted from (Woo et al., 2022).

The architecture is built upon three core principles:

1. **Modular Decomposition:** Each layer in the encoder progressively extracts latent growth and seasonal representations from the input time-series, while the level component is derived similarly to classical exponential smoothing methods.
2. **Emphasis on Recent Observations:** Inspired by exponential smoothing, the model assigns higher weights to recent observations, improving its capacity to model the trend component.
3. **Human Interpretability:** The final forecast is a composition of level, trend, and seasonal components, making the model's predictions interpretable and more suitable for practical applications.

The encoder is tasked with extracting growth and seasonality components from the lookback window. Unlike traditional Transformers, which rely on additive or multiplicative assumptions for seasonality, ETSformer uses residual learning to capture complex patterns (Woo et al., 2022). Each encoder layer processes the input signal sequentially, removing extracted growth and seasonal signals from the residual before passing it to the next layer.

Decoder Design

The decoder generates the final forecast by composing the extracted level, growth, and seasonal components over the forecast horizon. The forecast is defined as:

$$\hat{X}t{:}t+H = Et{:}t+H + \text{Linear}\Big(\sum_{n=1}^{N}(B^{(n)}t{:}t+H + S^{(n)}t{:}t+H)\Big).$$

Exponential Smoothing Attention (ESA)

The ESA mechanism replaces the conventional dot-product attention by leveraging a relative time-lag based approach. ESA assigns higher weights to recent observations, a characteristic that mimics the exponential decay seen in classical smoothing methods. This mechanism is formalized as:

$$\text{AES}(V)t = \alpha V_t + (1-\alpha)\text{AES}(V)t-1,$$

$$\text{AES}(V)t = \sum j=0^{t-1}\alpha(1-\alpha)^j V_{t-j} + (1-\alpha)^t v_0,$$

where α is the smoothing parameter, and v_0 is the initial state. This formulation, effectively captures short-term dependencies while maintaining computational efficiency with a complexity of $O(L\log L)$ (Woo et al., 2022).

Frequency Attention

The FA mechanism employs the Discrete Fourier Transform (DFT) to identify and extract dominant seasonal patterns in the input data. It selects the *K* largest amplitude bases from the frequency domain and transforms them back to the time domain to capture the seasonal component.

$$S_{j,i}^{(n)} = \sum_{k=1}^{K} A_{\varkappa(k),i}\Big[\cos\big(2\pi f_{\varkappa(k)}j + \Phi_{\varkappa(k),i}\big) + \cos\big(2\pi \bar{f}_{\varkappa(k)}j + \bar{\Phi}_{\varkappa(k),i}\big)\Big],$$

where $A_{\varkappa(k),i}$ and $\Phi_{\varkappa(k),i}$ are the amplitude and phase of the *k*-th frequency, and $f_{\varkappa(k)}$ is the Fourier frequency (Woo et al., 2022). This mechanism efficiently captures periodic patterns in time-series data.

Growth Damping

To enhance the model's long-term forecasting capability, ETSformer incorporates a growth damping mechanism similar to the concept of trend damping in exponential smoothing. The growth representation is defined as:

$$\mathrm{TD}\left(B_t^{(n)}\right)j = \sum i = 1^j \gamma^i B_t^{(n)},$$

where γ is a learnable damping parameter, controlling the rate at which the growth trend diminishes over time.

ETSformer demonstrates superior performance across multiple time-series benchmarks, significantly outperforming other models such as Autoformer and Informer, particularly in capturing complex seasonality and trends (Woo et al., 2022). The model's interpretability is enhanced through its decomposition into level, growth, and seasonal components, allowing users to visualize and understand the underlying patterns influencing forecasts (Woo et al., 2022).

Its ability to decompose time-series data into level, growth, and seasonal components, along with the novel ESA and FA mechanisms, makes it a powerful and interpretable model for forecasting complex temporal patterns (Woo et al., 2022).

3.4 Informer

The Informer model, proposed by H. Zhou et al., (2021) addresses critical challenges in long sequence time-series forecasting (LSTF) using Transformer-based architectures. Traditional Transformers struggle with quadratic time complexity, high memory consumption, and inefficiencies in the encoder-decoder architecture, which limits their application in LSTF (H. Zhou et al., 2021).

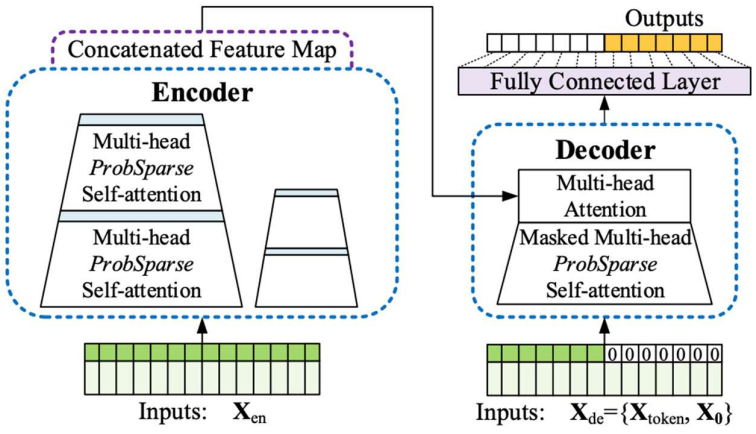


Figure 5 - Informer model overview. Reprinted from (H. Zhou et al., 2021).

H. Zhou et al., (2021) introduces three key innovations to overcome these issues: the ProbSparse self-attention mechanism, self-attention distilling, and a generative style decoder described as follow:

- **ProbSparse Self-Attention:** This mechanism reduces the time complexity from $O(L2)$ *to* $O(LlogL)$ by focusing only on a subset of queries that have the highest impact on the attention scores, significantly improving both efficiency and performance (H. Zhou et al., 2021). As illustrated in Figure 5, the model replaces the traditional self-attention mechanism in the encoder with the ProbSparse self-attention, significantly reducing the model's computational load and memory usage.
- **Self-Attention Distilling**: To further optimize memory usage, the model uses a cascading reduction in input sequence length, as depicted in Figure 6. This method privileges dominant attention scores, sharply reducing the input sequence dimension with each layer, thereby allowing the model to process extremely long sequences efficiently (H. Zhou et al., 2021)
- **Generative Style Decoder**: Unlike traditional step-by-step decoders, the Informer's decoder predicts long time-series sequences in a single forward operation (see Figure 5, right panel). This approach reduces inference time drastically and avoids the error propagation common in autoregressive models. This is visually represented in the right part of Figure 5, where the decoder structure is shown to predict long sequences in a generative manner without relying on previous time steps.

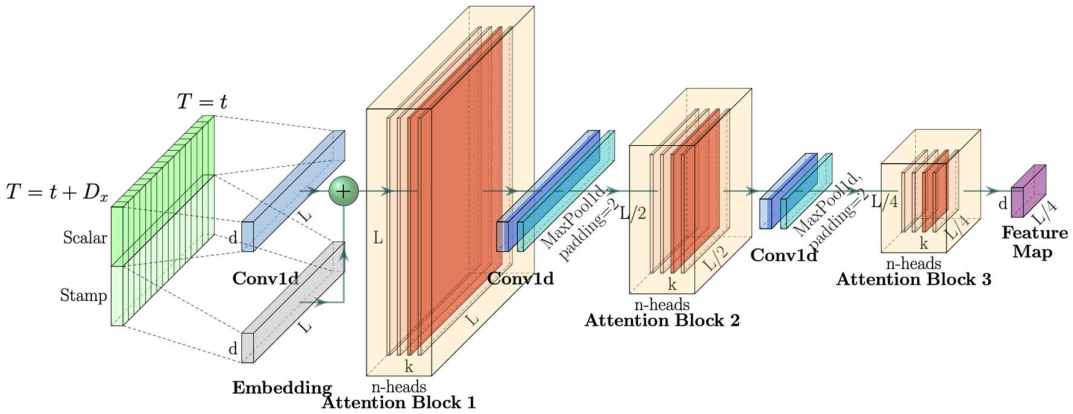


Figure 6 - The Informer single stack encoder. Reprinted from (H. Zhou et al., 2021).

3.5 NSTransformer

Non-stationary Transformers (NSTransformers) aim to improve the reliability and precision of forecasts for time series that exhibit non-stationary patterns by integrating novel mechanisms to effectively handle non-stationarity in time series data (Liu et al., 2022).

The NSTransformers framework, illustrated in Figure 7, introduces two complementary modules: Series Stationarization and De-stationary Attention. These modules work together to improve the predictability of non-stationary time series data while preserving the inherent non-stationary information that is crucial for accurate forecasting.

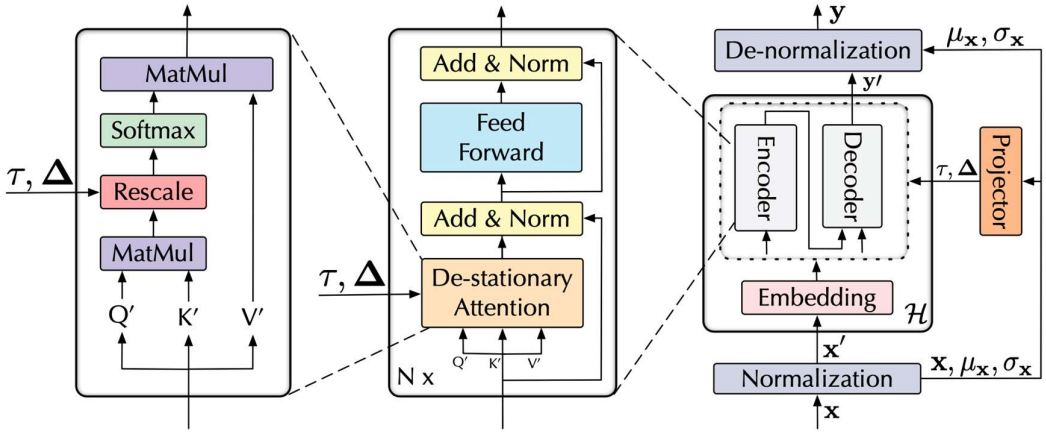


Figure 7 - Non-stationary Transformers. Reprinted from (Liu et al., 2022).

Series Stationarization

Series Stationarization is designed to mitigate the effect of non-stationarity by normalizing the time series data. This process involves transforming each input series $x = [x_1, x_2, ..., x_S]^\top$ into a stationarized version $x' = [x'_1, x'_2, ..., x'_S]^\top$ using the following normalization procedure:

$$\mu_x = \frac{1}{S}\sum_{i=1}^{S} x_i, \quad \sigma_x^2 = \frac{1}{S}\sum_{i=1}^{S}(x_i - \mu_x)^2,$$

where μ_x is the mean of the input series, calculated as the sum of all observations divided by the total number of observations S. The variance σ_x^2 is calculated as the average of the squared deviations from the mean.

The normalized series x'_i is obtained by subtracting the mean μ_x from each element of the series x_i and then dividing by the standard deviation σ_x, as shown below:

$$x'_i = \frac{1}{\sigma_x} \odot (x_i - \mu_x),$$

where $\odot$ denotes element-wise multiplication. This normalization step reduces the variability among different time series, making the data more amenable to modelling with deep learning architectures (Liu et al., 2022).

After the base model H forecasts the future values $y' = H(x')$, the model output is de-normalized to obtain the final prediction $\hat{y}$ using the following de-normalization formula:

$$\widehat{y_i} = \sigma_x \odot (y'_i + \mu_x).$$

This two-stage process ensures that the model inputs are stationarized, improving generalization, while the outputs are restored to their original statistical properties.

De-stationary Attention

While series stationarization improves the predictability of the input series, it may result in the loss of crucial non-stationary information, leading to over-stationarization. To address this issue, the De-stationary Attention mechanism is introduced. This mechanism restores the non-stationary characteristics of the original series into the temporal dependencies captured by the model. It achieves this by approximating the attention weights learned from the unstationarized data.

The standard self-attention mechanism in Transformers computes the attention score as:

$$\text{Attn}(Q, K, V) = \text{Softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V,$$

where Q, K, and V represent the query, key, and value matrices, respectively, and d_k is the dimensionality of the keys. In the De-stationary Attention mechanism, the attention scores are modified by incorporating de-stationary factors τ and Δ, learned from the original, unstationarized series x. The de-stationary factors are computed as follows:

$$\log \tau = \text{MLP}(\sigma_x, x), \quad \Delta = \text{MLP}(\mu_x, x),$$

where MLP represents a multi-layer perceptron that learns the de-stationary factors τ and Δ from the statistics of the original series. These factors are then used to adjust the standard attention computation:

$$\text{Attn}(Q', K', V', \tau, \Delta) = \text{Softmax}\left(\frac{\tau Q' K'^{\top} + 1\Delta^{\top}}{\sqrt{d_k}}\right)V',$$

where $Q'$and K' are the stationarized query and key matrices. These factors ensure that the model's attention weights remain sensitive to the intrinsic non-stationary properties of the data.

Non-stationary Transformers were evaluated on multiple real-world time series datasets, demonstrating superior performance over baseline models like Vanilla Transformer, Informer, and Reformer (Liu et al., 2022).

3.6 Reformer

The Reformer model, introduced by Kitaev et al., (2020), aims to address the high computational and memory costs associated with traditional Transformer models when applied to long sequences. While Transformers have achieved state-of-the-art results in various natural language processing tasks, their $O(L^2)$ time and space complexity, where L is the sequence length, limits their application to very long sequences. The Reformer tackles this limitation by introducing two key innovations: Locality-Sensitive Hashing (LSH) attention, which reduces the complexity of the attention mechanism, and Reversible Residual Layers, which reduce memory usage during training (Kitaev et al., 2020).

Locality-Sensitive Hashing Attention Mechanism

The standard attention mechanism in Transformers involves calculating the dot product between all query and key pairs, resulting in a complexity of $O(L^2)$. This becomes computationally infeasible for long sequences. The Reformer replaces the traditional attention with LSH attention, reducing the complexity to $O(L\log L)$ (Kitaev et al., 2020).

In the traditional Transformer, the attention mechanism is defined as:

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^{\top}}{\sqrt{d_k}}\right)V,$$

where $Q, K, and\ V$ are the query, key, and value matrices, respectively, and d_k is the dimension of the keys (Vaswani et al., 2017). The LSH attention mechanism optimizes this process by using locality-sensitive hashing to group similar queries and keys into the same hash bucket, significantly reducing the number of dot products that need to be computed. The LSH scheme assigns each vector x to a hash $h(x)$ such that similar vectors are likely to have the same hash. This is achieved by employing random projections as follows: $h(x) = \text{argmax}([xR; -xR])$ where R is a random matrix of size $[d_k, b/2]$ and $[u;v]$ denotes the concatenation of two vectors u and v (Andoni et al., 2015). LSH attention is then computed by restricting the set P_i of target items a query at position i can attend to, allowing attention only within the same hash bucket: $P_i = \{j : h(q_i) = h(k_j)\}$.

This approach significantly reduces the number of key-value pairs considered for each query, reducing both computation and memory requirements (Kitaev et al., 2020).

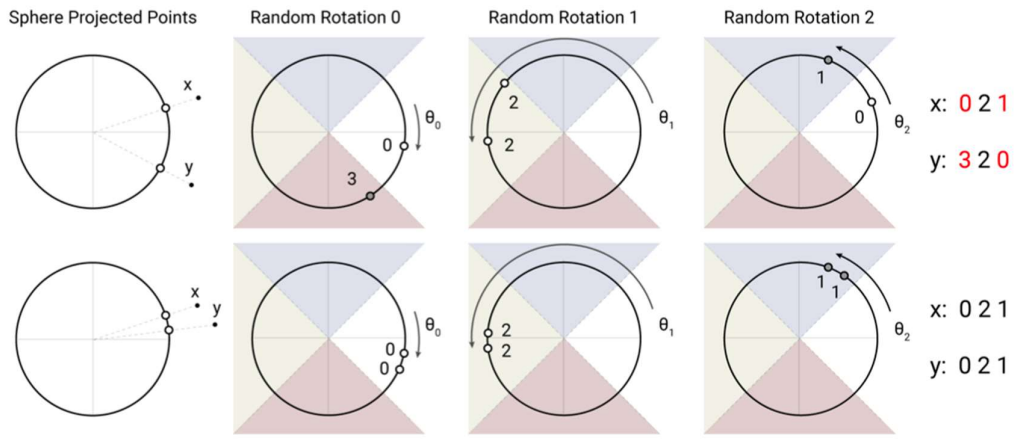


Figure 8 - LSH Attention Mechanism in Reformer. Reprinted from (Kitaev et al., 2020).

Multi-Round LSH Attention

To further improve accuracy, Reformer uses multiple rounds of hashing with distinct hash functions. For each round r, the attention is computed over the set:

$$P_i^{(r)} = \{j : h^{(r)}(q_i) = h^{(r)}(q_j)\}.$$

The final attention output is the weighted sum of the outputs from all rounds:

$$o_i = \sum_{r=1}^{n_{rounds}} \exp\left(z\left(i, P_i^{(r)}\right) - z\left(i, P_i\right)\right) o_i^{(r)},$$

where $z\left(i, P_i\right)$ is the normalizing term, and $o_i^{(r)}$ is the output of the r-th round of attention (Kitaev et al., 2020).

Reversible Residual Layers

Transformers typically store activations at each layer during training to facilitate backpropagation, resulting in high memory usage. Reformer addresses this issue by using reversible residual layers, inspired by RevNets (Gomez et al., 2017), where each layer's input can be reconstructed from its output, eliminating the need to store intermediate activations.

A standard residual layer updates its input x: $y = x + F(x)$.

In contrast, a reversible layer maintains two states, (x_1, x_2), and updates them as follows: $y_1 = x_1 + F(x_2)$, $y_2 = x_2 + G(y_1)$.

These transformations can be inverted to recover the original inputs x_1 and x_2:

$$x_1 = y_1 - F(x_2),\ x_2 = y_2 - G(y_1).$$

By applying this technique to Transformer layers, Reformer reduces memory usage during backpropagation without compromising model performance (Kitaev et al., 2020).

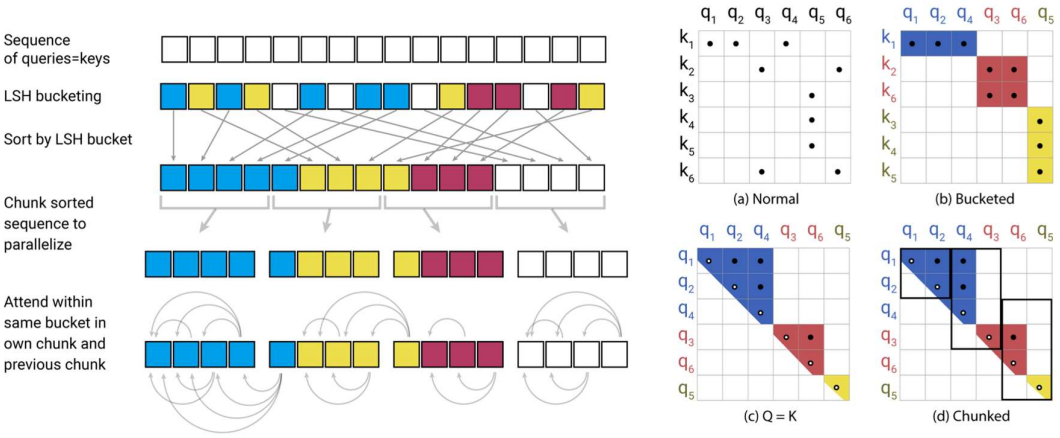


Figure 9 - Reversible Residual Layers in Reformer. Reprinted from (Kitaev et al., 2020).

Reformer was tested on various tasks, including text (enwik8) and image generation (imagenet-64), demonstrating comparable performance to traditional Transformers while significantly reducing memory and computation requirements (Kitaev et al., 2020). For example, on sequences of length 64K, Reformer outperformed the standard Transformer in terms of both speed and memory efficiency, allowing it to handle much longer sequences on standard hardware.

4 Empirical Study

This empirical study is designed to deeply evaluate the performance of advanced Transformer-based models for time series forecasting, approaching it not just as a technical exercise but as a meaningful investigation into how these models can serve real-world applications. Through systematic experimentation and comparative analysis, this study examines how these models stack up against traditional statistical methods and other modern deep learning techniques when applied to the demanding M5 competition dataset. We focus on six prominent Transformer-based architectures: Vanilla Transformer, Autoformer, ETSFormer, Informer, NSTransformer, and Reformer. The goal is to comprehensively understand their individual strengths and weaknesses, particularly in the domains of forecasting accuracy, computational efficiency, and robustness.

The M5 competition dataset (Howard et al., 2020) is an ideal choice for this evaluation, owing to its complexity and scale. It encompasses over 30,000 individual time series, derived from sales data across multiple product categories in different regions of the United States. This dataset presents a variety of temporal patterns - including seasonality, trends, and intermittency - which makes it an excellent benchmark for assessing the effectiveness of advanced forecasting models. The dataset allows us to simulate the real-life challenges faced in retail environments, such as high variability in consumer demand, multi-level aggregation requirements, and the computational burden of processing substantial volumes of data.

Our evaluation framework is methodically structured to provide a thorough understanding of each model's strengths and limitations. First, the Dataset (Section 4.1) is examined in detail, highlighting both the foundational sales data and the contextual features that are critical for training and validating the models. This section also delves into the preprocessing steps undertaken, such as normalization and feature engineering, which are essential for optimizing model performance. We then turn to Explanatory Variables (Section 4.2), emphasizing their crucial role in improving predictive accuracy. These explanatory variables provide additional context—such as time-related markers, product-specific characteristics, and special events—that helps the models capture external factors impacting sales, such as pricing strategies or seasonal holidays, thereby enhancing forecast quality.

In Hyperparameter Tuning (Section 4.3), we delve into the process of optimizing each model to achieve a balance between forecasting performance and computational efficiency. Hyperparameter tuning is a pivotal component of this empirical study, ensuring each model is tailored to the specific characteristics of the M5 dataset. This process involved a systematic exploration of a broad range of configurations, including context lengths, learning rates, and the number of encoder and decoder layers, using sophisticated optimization tools like Optuna. The insights gained from this process were instrumental in refining the models to reach an optimal trade-off between efficiency and accuracy, which is especially important given the dataset's scale and inherent variability.

We then outline the Performance Metrics (Section 4.4) used to evaluate the models. The metrics used include both traditional point forecasting metrics, such as Normalized Root Mean Squared Error and Root Mean Squared Error, and metrics specifically designed for probabilistic forecasting, such as Mean Weighted Quantile Loss and Mean Absolute Error Coverage. By using a combination of these metrics, we ensure a comprehensive evaluation that captures both the precision of point estimates and the ability of models to produce reliable probabilistic forecasts. This dual approach allows for a more nuanced evaluation, providing insight into how well each model aligns with actual observed values and how effectively it can quantify and manage uncertainty in its predictions.

The Results and Discussion (Section 4.5) synthesizes our findings, providing a detailed analysis of how each model performed across various metrics. We pay particular attention to identifying the models that excel at different forecasting horizons, and we discuss how hyperparameter settings and explanatory variables influenced each model's outcomes. Our findings reveal the trade-offs between model complexity, forecasting accuracy, and computational efficiency—insights that are crucial for assessing the practical feasibility of deploying these models in real-world, large-scale forecasting scenarios. Moreover, this section includes an analysis of model robustness across different temporal segments and product categories, offering a nuanced understanding of each model's strengths and remaining challenges. By adopting this comprehensive approach to evaluation and analysis, this study not only identifies the most effective models but also provides practical guidance on their deployment in dynamic, uncertain environments where forecasting accuracy is crucial.

4.1 Dataset

To ensure a robust evaluation of forecasting methods, this study leverages the well-known M5 competition dataset (Howard et al., 2020), which has been widely used as a benchmark in the field of time series forecasting (Makridakis et al., 2022b). The dataset is highly comprehensive, containing 30 490 time series derived from the sales data of 3 049 products across ten retail stores, representing different geographic locations in the United States - specifically California, Texas, and Wisconsin. The dataset spans approximately 5.4 years, from January 29, 2011, to June 19, 2016, resulting in 1,969 daily observations for each product.

A key characteristic of the M5 dataset is its hierarchical structure, which makes it particularly challenging and suitable for rigorous testing (Makridakis et al., 2022). The dataset encompasses three primary categories of products—Foods, Hobbies, and Household—each further divided into seven product departments. Sales records were collected daily, capturing the demand dynamics for different products in diverse retail environments.

Additionally, the extensive breadth of the M5 dataset makes it ideal for testing both point and probabilistic forecasting models, particularly those designed to handle large-scale, hierarchical, and heterogeneous time series data, as seen in retail contexts (Benidis et al., 2020). This dataset has been specifically chosen because of its high levels of volatility and intermittency, which provide an effective benchmark to evaluate the forecasting accuracy and robustness of advanced models such as Transformer-based architectures.

In this study, Transformer-based models, including the Vanilla Transformer, Autoformer, ETSformer, Informer, NSTransformer, and Reformer, are going to be evaluated using this dataset. The evaluation was conducted using a rolling cross-validation approach, employing a forecasting horizon of 28 days. This cross-validation process involved splitting the dataset into three evaluation windows, each spanning 28 days, to rigorously assess the model's performance across different forecast horizons. This methodology allows for a comprehensive understanding of each model’s forecasting capabilities, both in short and medium-term horizons, thus providing insight into their robustness and adaptability for real-world scenarios.

4.2 Explanatory Variables

In the empirical evaluation of time series forecasting, the use of explanatory variables is crucial for enhancing the predictive power of models. Explanatory variables help capture the underlying dynamics and dependencies in the data, providing additional context that improves the accuracy of both point and probabilistic forecasts.

The M5 dataset (Howard et al., 2020) used in this study includes several types of explanatory variables, covering time-related information, product-level characteristics, pricing information, and additional features such as special events. Table 1 provides a detailed summary of these variables and their impact on forecasting performance. These variables are summarized as follows:

- **Sales Lag Variables**: The dataset includes 30 lagged sales variables, such as lags of 1, 7, 30, and up to 1093 days. These lagged sales values are essential for capturing autoregressive patterns in time series and help models predict future sales based on historical sales dynamics (Taylor & Letham, 2018).
- **Time-related Variables**: Several time-related explanatory variables are included to capture seasonality and other time-related patterns. These variables consist of day of the week, day of the month, day of the year, month of the year, week of the year, week of the month, year, and an indicator for weekends. All these variables are encoded as categorical values and normalized. The inclusion of these features enables the models to capture recurring temporal patterns, such as increased sales on weekends or during specific holidays (Salinas et al., 2019).
- **Price Variables**: Price is a significant factor in demand forecasting. In the dataset, item price is normalized by the mean and standard deviation at three different aggregation levels—daily product price, department-level price, and store-level price. This normalization allows the model to effectively utilize pricing information in predicting sales by controlling for variations across different levels.
- **Event Variables**: Event variables include 31 different event names (such as Christmas, Halloween, or the Super Bowl) as well as event types categorized into cultural, national, religious, and sporting events. These variables are encoded and normalized to capture the impact of special events on consumer buying patterns. The influence of events is critical in predicting spikes or drops in sales for specific products, as certain holidays may drive increased demand in specific categories.

- **Product and Store Identifiers**: Categorical variables identifying products, departments, categories, stores, and states are also included. These variables are one-hot encoded or embedded, which allows the model to learn relationships between different products or geographic regions. This is particularly important in a retail context, where different products and stores exhibit different sales behaviors.

Data	No.of Covariates	Feature	Type	No. of categories	Encoding
Sales	30	Lags:{1, 2, 3, 4, 5, 6, 7, 8, 13, 14, 15, 20, 21, 22, 27, 28, 29, 30, 31, 56, 84, 363, 364, 365, 727, 728, 729, 1091, 1092, 1093}	Continuos	__	__
Time	9	Day of week Day of Month Day of Year Month of year Week of year Week of month Year	Categorical	7 31 366 12 53 6 6	Encoded as zero-based index and normalized to –[0.5, 0.5]
		Is weekend		2	Boolean
		Age	Continuous	__	$\log_{10}(2 + n$ sale days)
Price	3	Item's daily price normalized by mean/std Item's daily price normalized by department's daily mean/std price Item's daily price normalized by store's daily mean/std price	Continuous	__	
Snap	3	Supplemental nutrition assistance program days in CA, TX, WI	Categorical	3	Boolean
Events	2	Event name:{nan,ChanukahEnd,Christmas,CincoDeMayo,ColumbusDay,Easter,EidAlFitr,EidAlAdha,Father'sDay,Halloween,IndependenceDay,LaborDay,LentStart,LentWeek2,MartinLutherKingDay,MemorialDay,Mother'sDay,NBAFinalsEnd,NBAFinalsStart,NewYear,OrthodoxChristmas,OrthodoxEaster,PesachEnd,PresidentsDay,PurimEnd,RamadanStart,StPatricksDay,SuperBowl,Thanksgiving,ValentinesDay,VeteransDay}	Categorical	31	Encoded as zero-based index and normalized to –[0.5, 0.5]
	2	Event type: {nan,Cultural,National,Religious,Sporting}	Categorical	5	
ID	60	item id dept id cat id store id state id	Categorical	3049 7 3 3 3	One-hot encoded and embedded with min(50, n_categ+1)//2) dimension

Table 1 - Explanatory variables.

4.3 Hyperparameter Tuning

To optimize each model, hyperparameters were tuned with a focus on accuracy and computational efficiency. The process of hyperparameter tuning was aimed at identifying the optimal balance between performance and computational cost, using Mean Weighted Quantile Loss (MWQL) as the primary metric to gauge model effectiveness. Table 2 - Model's Hyperparameter Search Spaces Used in HPO, Table 3 - Hyperparameter Optimization Settings and Table 4 - Parameter configuration for each Transformer-based model, detail the range of hyperparameters considered, providing insights into the experimental setup for each Transformer-based model.

Hyperparameter	Range	
	Vanilla Transformer, Autoformer, Informer, NSTransformer, Reformer	**ETSformer**
Context length	{28, 28 ×2, 28 ×3}	{28, 28 ×2, 28 ×3}
Batch size	{32, 64, 128, 256}	{32, 64, 128, 256}
Number of enconder layers	{2, 4, 8, 16}	
Number of decoder layers	{2, 4, 8, 16}	

Table 2 - Model's hyperparameter search spaces used in HPO.

Parameter	**Value**
Number of trials	10
Number of epochs	10
Number of batches per epoch	50
Number of samples	20
Validation function	Mean Weighted Quantile Loss (MWQL)

Table 3 - Hyperparameter optimization settings for Transformer-based models.

The hyperparameters tested included context length, batch size and number of encoder and decoder layers. Different configurations were evaluated to understand their influence on model performance:

- Context Length and Batch Size: Hyperparameter tuning involved experimenting with context lengths of 28, 56, and 84 days, and batch sizes ranging from 32 to 256. As shown in Table 5, the optimal configuration for most models was a context length of 56 days and a batch size of 64. This balance enabled models to effectively capture the temporal dependencies while maintaining computational feasibility. Larger context lengths did not always lead to improved performance, as they sometimes introduced noise that diminished the model's predictive capability.
- Number of Layers: The depth of the encoder and decoder layers was varied from 2 to 16. The optimal number of layers differed across models but generally fell between 4 and 8, as summarized in Table 5. For example, the NSTransformer and Autoformer models achieved optimal performance with 4 encoder layers, when features are included in the model, striking a balance between model complexity and the risk of overfitting (Goodfellow et al., 2016). Models with deeper layers, such as 16, tended to overfit the training data, particularly in cases with high levels of noise.

Optimal hyperparameter configurations were ultimately selected based on a trade-off between minimizing forecast errors and computational efficiency. The hyperparameter optimization results, presented in Table 5 - Optimal Hyperparameter Configurations from Optuna, provide a detailed overview of the settings that led to the highest performance for each model. These settings were instrumental in fine-tuning each model to maximize both accuracy and efficiency, ensuring that the models are viable for large-scale retail forecasting applications, which often require quick retraining and real-time forecasting capabilities (Bengio, 2012).

The hyperparameter tuning process highlighted the importance of balancing model complexity with computational efficiency. While deeper and more complex models have the potential to capture intricate temporal dependencies, they also risk overfitting and increased training times. Therefore, the optimal configurations identified in this study aimed to maximize accuracy while ensuring that the computational requirements remained practical for real-world applications.

Parameter	Vanilla Transformer	Autoformer	ETSformer	Informer	NSTransformer	Reformer
Prediction length of decoder	28					
Distribution output	Student's t					
Loss function	Negative log likelyhood					
Size of target	1					
Scale of the input target	Mean	Std	Std	Std	-	Std
Lags sequence	[1, 2, 3, 4, 5, 6, 7, 8, 13, 14, 15, 20, 21, 22, 27, 28, 29, 30, 31, 56, 84, 363, 364, 365, 727, 728, 729, 1091, 1092, 1093]					
Dimensionality of Transformer layers	32	-	64	64	-	64
Number of attention heads	2					
Feedforward hidden size	32	32	-	32	32	-
Activation function	gelu	relu	-	relu	gelu	-
Dropout for fully connected layers	0.1					
Moving average window	-	25	-	-	-	-
Autocorrelation factor	-	1	-	-	-	-
Number of layers	-	-	2	-	-	-
K largest amplitudes	-	-	4	-	-	-
Embedding kernel size	-	-	3	-	-	-
Attention in encoder	-	-	-	ProbAttention	-	-
Use distilling in encoder	-	-	-	True	-	-
ProbSparse sampling factor	-	-	-	5	-	-

Table 4 - Parameter configuration for each Transformer-based model.

Hyperparameter	Vanilla Transformer	Autoformer	ETSformer	Informer	NStransformer	Reformer
			Without features			
Context length	28	28 x 2	28	28 x 3	28 x 3	28 x 2
Batch size	128	64	128	256	256	64
Number of encoder layers	16	16	-	16	4	4
Number of decoder layers	2	8	-	16	8	4
MWQL	0.6121	0.7403	0.6178	0.7753	0.6081	1.5830
			With features			
Context length	28	28 x 3	28	28 x 2	28 x 3	28
Batch size	256	256	128	128	256	32
Number of encoder layers	8	4	-	2	4	16
Number of decorder layers	4	4	-	2	4	2
MWQL	0.6067	0.7387	0.6312	0.7730	0.6228	1.3990

Table 5 - Optimal hyperparameter configurations from Optuna.

4.4 Performance Metrics

To evaluate the effectiveness of the Transformer-based models used in this study, several performance metrics were employed. The selection of these metrics was made to comprehensively capture both the accuracy of point forecasts and the reliability of probabilistic forecasts, which are crucial for understanding the different aspects of time series prediction. The metrics used include Normalized Root Mean Squared Error (NRMSE), Mean Absolute Scaled Error (MASE), Mean Weighted Quantile Loss (MWQL) and Mean Absolute Error (MAE) Coverage, with detailed results summarized in Tables 6 to 8.

Mean Squared Error (MSE) is a widely used metric that measures the average squared difference between actual and predicted values:

$$MSE = \frac{1}{n}\sum_{t=1}^{n}(y'_t - y_t)^2$$

where n is the length of the observed time series y.

MSE is particularly useful for penalizing larger errors more severely, which can help in identifying models that make substantial prediction mistakes.

Root Mean Squared Error (RMSE) is the square root of MSE, providing an error measure that is on the same scale as the original data:

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y'_t - y_t)^2}$$

RMSE is intuitive and commonly used to evaluate forecasting accuracy, particularly when larger errors are of greater concern.

Normalized Root Mean Squared Error (NRMSE) further extends the interpretation of RMSE by normalizing it using a measure such as the range or mean of the observed data:

$$NRMSE = \frac{RMSE}{mean(y)}$$

This normalization makes it possible to compare model performance across datasets with different scales, offering a scale-independent error metric. In this study, NRMSE was employed to ensure that the models were evaluated consistently across different data segments.

Mean Absolute Scaled Error (MASE) is a scale-independent metric that evaluates the accuracy of forecasts by comparing them to a naive baseline model:

$$MASE = \frac{\frac{1}{h}\sum_{t=n+1}^{n+h} |y'_t - y_t|}{\frac{1}{n-1}\sum_{t=2}^{n} |y_t - y_{t-1}|}$$

where n is the length of the observed time series y and h is the length of the forecast horizon (Hyndman & Koehler, 2006).

MASE is particularly effective for comparing models across different datasets, as it is invariant to scaling. The use of MASE also highlighted the importance of capturing both short-term and long-term temporal dependencies, as models that struggled with one or the other tended to have higher MASE values.

Mean Weighted Quantile Loss (MWQL) is used to assess the quality of probabilistic forecasts by evaluating how well a model predicts various quantiles of the future distribution. This is particularly important for applications that require an understanding of uncertainty, such as inventory management, where both overestimation and underestimation carry significant costs.

For a set of N quantiles $\{q_1,\ldots,q_N\}$, the Mean Weighted Quantile Loss is defined as:

$$MWQL = \frac{1}{N\sum_{t=T+1}^{T+h}|y_t|}\sum_{t=T+1}^{T+h}\sum_{j=1}^{N}\rho_{q_j}(y_t, f_t^{q_j})$$

where N is the number of quantiles, T is the length of the observed time series, h is the length of the forecast horizon, y_t is the actual value for observation t of time series y, $f_t^{q_j}$ is the predicted quantile q_j of time series y at time t and $\rho_{q_j}(y_t, f_t^{q_j})$ is the quantile loss at level q_j, which is defined as:

$$\rho_{q_j}(y_t, f_t^{q_j}) = \begin{cases} 2(1-q_j)(f_t^{q_j} - y_t), if\, y_t < f_t^{q_j} \\ 2q_j(y_t - f_t^{q_j}), if\, y_t \geq f_t^{q_j} \end{cases}.$$

The Mean Absolute Error Coverage (MAEC) is a metric used to assess the quality of probabilistic forecasts. It quantifies the proportion of time points where the actual value lies below the predicted quantile. For a set of N quantiles $\{q_1,\ldots,q_N\}$, the MAEC is defined as:

$$MAEC = \frac{1}{N}\sum_{j=1}^{N}|Coverage_{q_j} - q_j|$$

where N is the number of quantiles and $Coverage_{q_j}$ is defined as:

$$Coverage_{q_j} = \frac{1}{h}\sum_{t=T+1}^{T+h}\tau_{q_j}(y_t, f_t^{q_j})$$

where T is the length of the observed time series, h is the length of the forecast horizon, y_t is the actual value for observation t of time series y, $f_t^{q_j}$ is the predicted quantile q_j of time series y at time t and $\tau_{q_j}(y_t, f_t^{q_j})$ is defined as:

$$\tau_{q_j}(y_t, f_t^{q_j}) = \begin{cases} 1, if\, y_t \leq f_t^{q_j} \\ 0, if\, y_t > f_t^{q_j} \end{cases}.$$

MAEC primarily assesses how well the prediction intervals cover the true values. A lower MAEC value indicates better coverage, meaning the prediction intervals are more likely to contain the actual values.

These four metrics were selected to provide a balanced assessment of both the point forecasting capabilities and the probabilistic forecasting aspects of each

model. NRMSE provides insight into the overall accuracy, with a specific focus on penalizing large deviations, while MASE offers a way to understand forecast performance in a scale-independent manner. Meanwhile, MWQL and MAEC allow for evaluating the effectiveness of the model in scenarios that require accurate estimates of uncertainty.

4.5 Results and Discussion

The empirical evaluation conducted in this study aimed to assess the performance of various Transformer-based models for time series forecasting, including the Vanilla Transformer, Autoformer, ETSFormer, Informer, NSTransformer, and Reformer, both with and without external information. This section provides a comprehensive discussion of the results obtained, focusing on the strengths and weaknesses of each model. The results are presented in Tables 6 to 8 and reveal important insights into the comparative performance of these models across different evaluation metrics and forecast horizons. By exploring both point forecast and probabilistic forecast metrics, we can better understand the suitability of each model for different types of time series forecasting applications.

Table 6 presents the evaluation of point forecast metrics (MASE and NRMSE), and the evaluation of probabilistic forecast metrics (MWQL and MAE Coverage), over the full forecast horizon (28 days). These metrics provide insights into the accuracy and consistency of each model when predicting future values, with MASE offering a scale-independent comparison and NRMSE focusing on the error magnitude in the context of the observed data scale.

Model		Point forecast metrics		Probabilistic forecast metrics				
		MASE	NRMSE	WQL0.1	WQL0.5	WQL0.9	MWQL	MAE Coverage
Vanilla Transformer	Without features	**0.902**	1.748	0.254	**0.735**	0.675	**0.629**	0.081
	With features	**0.906**	**1.650**	**0.227**	**0.738**	0.716	**0.634**	0.190
Autoformer	Without features	1.062	2.421	0.378	0.859	0.701	0.741	0.070
	With features	1.054	2.439	0.395	0.925	0.731	0.796	**0.071**
ETSformer	Without features	0.985	1.984	0.316	0.769	**0.595**	0.646	**0.049**
	With features	1.067	1.738	0.328	0.797	**0.615**	0.674	0.127
Informer	Without features	1.026	2.470	0.322	0.921	0.935	0.801	0.109
	With features	0.996	2.522	0.253	0.942	1.038	0.830	0.086
NSTransformer	Without features	0.917	**1.669**	**0.249**	0.746	0.680	0.636	0.073
	With features	0.979	1.849	0.263	0.785	0.781	0.687	0.178
Reformer	Without features	2.463	4.550	0.547	2.217	3.182	2.097	0.485
	With features	1.979	3.892	0.468	1.768	2.329	1.642	0.449

Table 6 - Result metrics over full horizon.

The Vanilla Transformer model demonstrated competitive performance across both point and probabilistic forecasting metrics. For point forecasting, the MASE increased slightly from 0.902 without features to 0.906 with features, indicating that incorporating features did not lead to a significant improvement in accuracy. However, the NRMSE decreased from 1.748 to 1.650 with the inclusion of

features, suggesting that the model was able to better capture overall trends with additional context. Regarding probabilistic forecasting, the MWQL increased slightly from 0.629 to 0.634 with features, indicating a minor increase in forecast uncertainty. Despite this, the MAE Coverage improved significantly from 0.081 to 0.190 with features, highlighting that additional context allowed the model to provide more comprehensive prediction intervals, which is crucial for effective uncertainty estimation (Howard et al., 2020).

The Autoformer model showed mixed results, regarding the point forecasting, the MASE decreased from 1.062 to 1.054 with features, suggesting a marginal improvement from the inclusion of contextual information. However, the NRMSE increased from 2.421 to 2.439, indicating that while features may have helped with general accuracy, they also introduced complexities that the model struggled to handle effectively. In terms of probabilistic metrics, the MWQL increased from 0.741 to 0.796, implying increased uncertainty in the forecasts, and the MAE Coverage increased only slightly from 0.070 to 0.071, indicating limited benefit in capturing prediction uncertainty from the inclusion of features (Makridakis et al., 2022)

ETSFormer also exhibited mixed performance when considering both point and probabilistic metrics. For point forecasting, the MASE increased from 0.985 to 1.067 with features, which could imply overfitting or difficulties in generalizing with added complexity. However, the NRMSE improved from 1.984 to 1.738, showing that the model was more capable of reducing large errors with features. Regarding probabilistic forecasting, the MWQL increased slightly from 0.646 to 0.674, while the MAE Coverage improved significantly from 0.049 to 0.127. These results highlight the potential benefit of feature complexity in improving uncertainty estimation, although caution is needed to avoid overfitting (Benidis et al., 2020).

The Informer model demonstrated an overall improvement with the inclusion of features, the MASE decreased from 1.026 to 0.996, and the NRMSE increased slightly from 2.470 to 2.522, indicating that additional contextual data didn’t improved the model's ability to capture temporal patterns effectively. Regarding the MWQL increased slightly from 0.801 to 0.830 with features, while the MAE Coverage decreased from 0.109 to 0.086. This indicates that Informer provides limited uncertainty estimates, making it suitable for scenarios requiring high reliability in point forecasting (Wu et al., 2021).

NSTransformer model demonstrated consistent performance, particularly when additional features were included, the MASE increased slightly from 0.917 to 0.979, and the NRMSE increased from 1.669 to 1.849, indicating some degradation in accuracy. However, NSTransformer still maintained a high level of robustness, adapting well to different scenarios despite added feature complexity. Regarding probabilistic metrics, the MWQL increased slightly from 0.636 to 0.687, while the MAE Coverage showed significant improvement from 0.073 to 0.178, suggesting that the model's reliability in uncertainty estimation benefited from the additional features (Zhang & Yan, 2022).

Reformer was the weakest performer among all models evaluated. The MASE improved from 2.463 to 1.979 with features, and the NRMSE decreased from 4.550 to 3.892. Although there was some improvement, these values were still considerably higher than those of the other models, highlighting Reformer's struggle to capture the temporal complexity inherent in the data. For probabilistic forecasting, the MWQL decreased from 2.097 without features to 1.642 with features, reflecting some reduction in forecast uncertainty, yet it remained significantly higher compared to other models. The MAE Coverage also decreased slightly from 0.485 to 0.449, suggesting limited improvement in the quality of uncertainty estimation despite the addition of features (Kitaev et al., 2020).

Tables 7 and 8 provide a detailed comparison of the performance of the six Transformer-based models in point and probabilistic forecasting tasks. The models are evaluated across four forecast horizons (1, 7, 14, and 21 steps ahead) and two scenarios: without and with the incorporation of external features. The evaluation metrics used are Mean Absolute Scaled Error (MASE) and Normalized Root Mean Squared Error (NRMSE) for point forecasting and Mean Weighted Quantile Loss (MWQL) and Mean Absolute Error Coverage (MAEC) for probabilistic forecasting.

Model		MASE				NRMSE			
		1 step	7 steps	14 steps	21 steps	1 step	7 steps	14 steps	21 steps
Vanilla Transformer	Without features	0.836	0.855	0.886	0.898	1.488	1.639	1.659	1.704
	With features	0.837	0.856	0.888	0.903	1.383	1.535	1.567	1.615
Autoformer	Without features	1.027	1.043	1.065	1.067	2.408	2.530	2.469	2.452
	With features	1.001	1.024	1.054	1.059	2.112	2.442	2.418	2.431
ETSformer	Without features	0.919	0.945	0.974	0.985	1.859	2.129	1.986	1.987
	With features	0.988	1.021	1.053	1.066	1.534	1.663	1.684	1.709
Informer	Without features	0.981	1.004	1.023	1.029	2.227	2.487	2.448	2.446
	With features	0.879	0.915	0.968	0.991	2.152	2.435	2.473	2.496
NSTransformer	Without features	0.857	0.872	0.902	0.916	1.402	1.543	1.589	1.634
	With features	0.871	0.904	0.948	0.969	1.565	1.775	1.792	1.829
Reformer	Without features	1.621	1.873	2.151	2.334	2.908	3.521	3.943	4.277
	With features	1.379	1.585	1.786	1.910	2.881	3.331	3.575	3.754

Table 7 - Results of point forecasting over differing forecast horizon.

The Vanilla Transformer model exhibited some of the lowest MASE and NRMSE scores across all forecast horizons, indicating that it is one of the best models in terms of forecasting accuracy and error management. Its performance was particularly stable, with only minor increases in error metrics as the forecast horizon extended from 1 step to 21 steps, highlighting its reliability for both short-term and long-term predictions. Similarly, the NSTransformer also consistently achieved lower MASE values compared to other models, making it one of the best performers. This model showed an ability to produce accurate forecasts with minimal error over different forecast horizons, which implies strong adaptability to various temporal dynamics in the time series.

The ETSformer model also performed quite well, particularly for NRMSE, which remained relatively low across different forecast horizons, emphasizing its strength in managing error variability. It consistently stayed competitive against other models, demonstrating an ability to maintain forecast stability over time, which is critical for long-term forecasting accuracy. Another strong performer was Informer, which showed relatively low NRMSE and MASE values across different horizons, making it a reliable option for reducing forecast errors effectively.

On the other hand, the Reformer model consistently had the highest MASE and NRMSE values among all models, indicating that it performed the worst in terms of forecast accuracy and error minimization. Its MASE and NRMSE values significantly increased as the forecast horizon extended, which indicates that this model struggles to manage time series dependencies effectively over longer horizons. This poor scalability over time makes Reformer an unsuitable choice for situations that require accurate long-term forecasts. Additionally, the Autoformer model showed higher MASE and NRMSE

values compared to other models like Vanilla Transformer, NSTransformer, and ETSformer. Its performance was less effective in capturing the underlying patterns of the time series, especially for extended forecast horizons of 14 to 21 steps, suggesting its limitations in adapting well to long-term prediction tasks.

The models with lower MASE and NRMSE values - such as Vanilla Transformer, NSTransformer, and ETSformer - emerged as the best models for point forecasting without additional features. These models demonstrated a clear capability to produce accurate forecasts with minimal error, making them highly reliable for both short and long-term forecasts. On the contrary, Reformer was the worst performing model across all forecast horizons, with consistently high error metrics that make it unsuitable for practical forecasting applications. Therefore, Vanilla Transformer is highly recommended as the top-performing model due to its consistent accuracy, while Reformer should be reconsidered or avoided due to its high error rates and weak ability to handle longer forecast horizons effectively.

Model		**MWQL**				**MAE Coverage**			
		1 step	7 steps	14 steps	21 steps	1 step	7 steps	14 steps	21 steps
Vanilla Transformer	Without features	0.594	0.588	0.605	0.618	0.056	0.061	0.070	0.072
	With features	0.585	0.587	0.605	0.622	0.083	0.133	0.155	0.176
Autoformer	Without features	0.761	0.744	0.739	0.740	0.051	0.060	0.064	0.067
	With features	0.790	0.787	0.789	0.794	0.069	0.067	0.069	0.070
ETSformer	Without features	0.623	0.621	0.632	0.641	0.046	0.046	0.048	0.049
	With features	0.645	0.646	0.658	0.668	0.126	0.124	0.128	0.128
Informer	Without features	0.783	0.796	0.794	0.798	0.095	0.097	0.105	0.108
	With features	0.711	0.737	0.780	0.814	0.059	0.065	0.076	0.083
NSTransformer	Without features	0.595	0.594	0.613	0.627	0.043	0.050	0.061	0.068
	With features	0.608	0.632	0.650	0.671	0.058	0.096	0.137	0.161
Reformer	Without features	1.283	1.489	1.753	1.949	0.423	0.455	0.473	0.481
	With features	1.133	1.292	1.448	1.562	0.339	0.388	0.420	0.438

Table 8 - Results of probabilistic forecasting over differing forecast horizon.

In terms of probabilistic forecasting, the Vanilla Transformer performs consistently across horizons, with slightly lower MWQL and moderate improvements in MAE Coverage when features are included. This suggests the model benefits from features, although the improvements are relatively minor, especially at longer horizons. The Autoformer, while not showing a drastic improvement with features, maintains lower MWQL values than many other models, particularly at shorter horizons, and has stable MAE Coverage that marginally increases at the 21-step horizon, indicating its reliability with features. ETSformer also benefits from features, displaying slightly lower MWQL

across horizons and a significant increase in MAE Coverage at shorter horizons, making it one of the stronger models overall with features.

Informer sees notable improvements with features, especially in MWQL, which drops considerably, and MAE Coverage, which increases substantially at shorter horizons. This indicates that Informer's performance is more responsive to feature inclusion and is well-suited for short- to medium-term forecasts. NSTransformer also benefits from features, with gradual increases in MAE Coverage and consistently low MWQL across horizons, showing it is effective for both short and longer horizons. In contrast, Reformer performs the poorest among the models, with both high MWQL and low MAE Coverage, especially without features. Its performance remains weak even with feature inclusion, suggesting it is not well-suited for the task, particularly at longer horizons.

In contrast, Reformer performs the poorest among the models, with both high MWQL and low MAE Coverage, especially without features. Its performance remains weak even with feature inclusion, suggesting it is not well-suited for the task, particularly at longer horizons.

The following set of charts provides a comprehensive visual comparison of the performance of all models in terms of different metrics across varying forecast horizons (1, 7, 14, and 21 steps), according to the results in the Table 7 and Table 8.

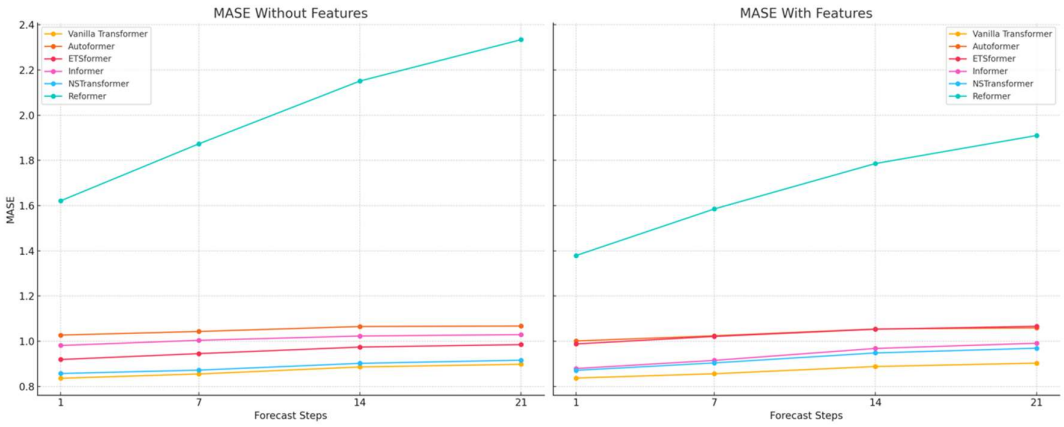


Figure 10 - MASE scores for different Transformer-based model, without features and with features, at various forecast horizons

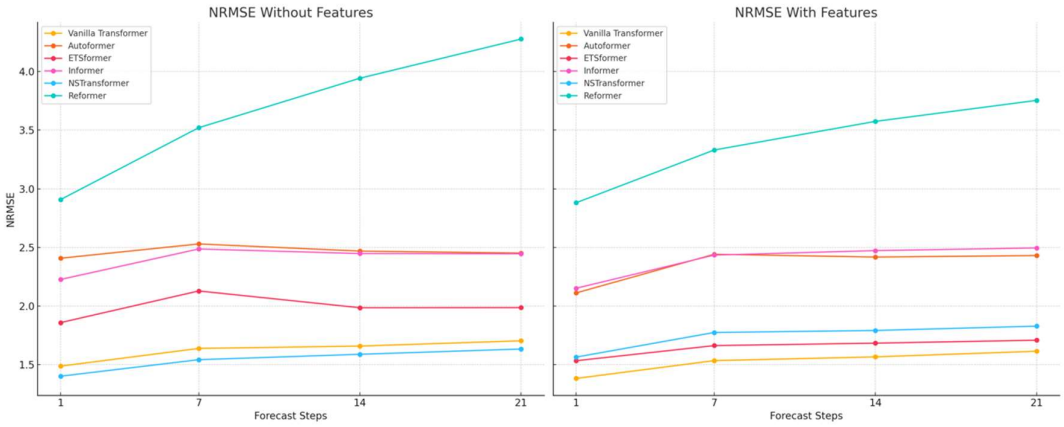


Figure 11 - NRMSE scores for different Transformer-based model, without features and with features, at various forecast horizons

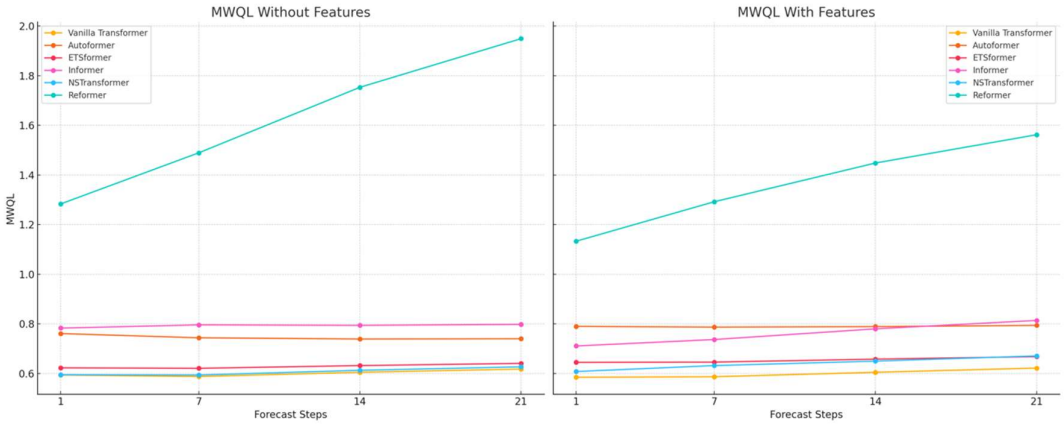


Figure 12 - MWQL scores for different Transformer-based model, without features and with features, at various forecast horizons

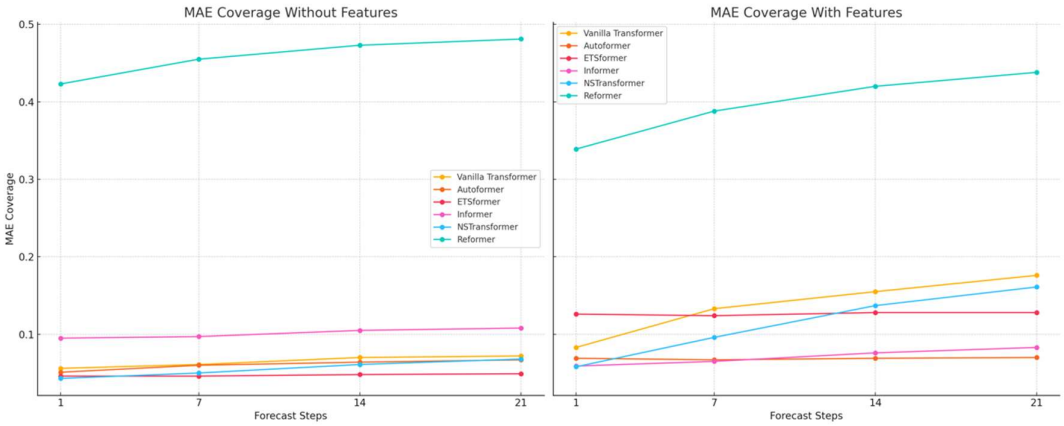


Figure 13 – MAE Coverage scores for different Transformer-based model, without features and with features, at various forecast horizons

5 Conclusion

The empirical evaluation conducted in this study provides significant insights into the strengths, weaknesses, and practical feasibility of Transformer-based models for time series forecasting. This study compared six prominent models—Vanilla Transformer, Autoformer, ETSFormer, Informer, NSTransformer, and Reformer—by rigorously applying them to the challenging M5 competition dataset. The overarching goal was to determine their effectiveness in terms of forecasting accuracy, computational efficiency, and robustness, especially in large-scale and high-variability retail environments (Howard et al., 2020; Makridakis et al., 2022).

The results revealed that Transformer-based models offer notable advancements in time series forecasting, particularly for complex, multi-level data like the M5 dataset (Makridakis et al., 2022). Among the models evaluated, the Vanilla Transformer, NSTransformer, and ETSFormer consistently outperformed others, demonstrating lower error rates across different metrics and maintaining stability over extended forecast horizons. These models effectively balanced accuracy and computational efficiency, suggesting their suitability for both short - and long-term retail forecasting scenarios (Zhang & Yan, 2022). The NSTransformer and ETSFormer models stood out due to their adaptability, showing strong performance in capturing temporal dependencies across different forecast horizons. On the other hand, Reformer struggled significantly with the complexity and variability of the dataset, resulting in higher error metrics compared to the other models. Its limited ability to capture temporal dependencies over long forecast horizons indicates that it may be unsuitable for large-scale retail forecasting tasks, particularly when accurate long-term predictions are required. The Autoformer also showed limitations, particularly in handling long-term dependencies effectively, which suggests that further improvements are needed for these models to become more competitive (Kitaev et al., 2020).The study also highlighted the importance of hyperparameter tuning, demonstrating how configurations such as context length, batch size, and the number of encoder and decoder layers directly impact model performance. Specifically, the optimal trade-off between model complexity and computational feasibility was essential to achieving high forecasting accuracy while ensuring practical application in real-world settings (Goodfellow et al., 2016).

The inclusion of explanatory variables played a critical role in enhancing model performance. Features such as product-level characteristics, pricing information, and special events contributed to improving the models' capacity to capture underlying temporal patterns and provided better probabilistic forecasts (Salinas et al., 2019). However, it was observed that incorporating additional features did not always lead to significant improvements; in some cases, it introduced complexity that models like Reformer and Autoformer struggled to manage effectively.

Overall, this study underscores the potential of Transformer-based models to improve forecasting accuracy in dynamic and uncertain environments. The Vanilla Transformer, NSTransformer, and ETSFormer emerged as the top-performing models, balancing accuracy, efficiency, and robustness. These models provide promising avenues for further exploration and deployment in real-world forecasting applications, particularly in the retail sector where accurate demand forecasting is critical for operational efficiency. Future research could focus on further refining hyperparameter optimization and exploring hybrid models that combine the strengths of multiple Transformer-based architectures to enhance both accuracy and generalizability.

Implications for the Retail Industry

The results of this study offer important insights for the retail industry, which depends on accurate forecasting to manage inventory, optimize supply chains, and support strategic decision-making. The performance of models like the Vanilla Transformer, NSTransformer, and ETSFormer in capturing complex temporal patterns suggests that these models can be instrumental in helping retailers anticipate demand more effectively, thereby improving operational efficiency. Retailers can leverage these models to respond swiftly to changing market conditions, optimize promotional strategies, and enhance customer satisfaction through better product availability (Howard et al., 2020).

Another key implication lies in the use of explanatory variables, which were shown to significantly enhance model performance. Retailers should prioritize the collection and integration of diverse contextual data, including pricing, promotions, and external events, to fully leverage the predictive power of these advanced models (Makridakis et al., 2022).

Limitations

Despite promising outcomes, this study has limitations that should be considered. Transformer-based models are computationally intensive, requiring significant resources, which may pose challenges for smaller organizations without advanced infrastructure. The reliance on high-quality data is another limitation, as data inconsistencies or missing values can negatively impact model performance. Moreover, while the M5 dataset provided a comprehensive benchmark, the generalizability of the results to other domains or geographic regions remains uncertain. Lastly, hyperparameter tuning was found to be a complex process, requiring considerable expertise, which may limit the accessibility of these models to organizations without specialized knowledge.

Future Research Directions

Future research could explore several areas to build upon the findings of this thesis. Hybrid modeling approaches that combine Transformer-based architectures with traditional statistical models could offer both improved accuracy and enhanced interpretability (Hyndman & Athanasopoulos, 2021). Developing methods for automated hyperparameter optimization, such as AutoML, could make these advanced models more accessible (Feurer et al., 2015). Additionally, further research into explainability and interpretability techniques, such as attention visualization or integrating explainable AI methods, could improve the transparency of these models, making them more trustworthy for stakeholders (Doshi-Velez & Kim, 2017).

Exploring the application of these models in real-time forecasting environments would extend their practical utility, particularly in dynamic retail contexts where rapid adjustments are necessary. Finally, investigating the cross-domain adaptability of these models could validate their use in other fields, such as energy demand or traffic flow prediction, further broadening their applicability.

In conclusion, this study demonstrates the significant potential of Transformer-based models in advancing the field of time series forecasting, particularly within the retail industry. Among the models studied, the Vanilla Transformer, NSTransformer, and ETSFormer showcased superior performance in terms of accuracy, efficiency, and adaptability. The findings highlight that careful hyperparameter tuning, the inclusion of appropriate explanatory variables, and model selection are critical factors in achieving optimal forecasting outcomes. Despite limitations such as dataset specificity and

computational constraints, the insights gained provide a strong foundation for future exploration and innovation. With further advancements in model interpretability, hyperparameter optimization, and hybrid approaches, Transformer-based models can play a pivotal role in driving more accurate and efficient demand forecasting, ultimately benefiting the retail industry and beyond.

Andoni, A., Indyk, P., Laarhoven, T., Razenshteyn, I., & Schmidt, L. (2015). *Practical and Optimal LSH for Angular Distance* (arXiv:1509.02897). arXiv. https://doi.org/10.48550/arXiv.1509.02897

Bai, S., Kolter, J. Z., & Koltun, V. (2018, March 4). *An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling*. arXiv.Org. https://arxiv.org/abs/1803.01271v2

Bengio, Y. (2012). Deep Learning of Representations for Unsupervised and Transfer Learning. *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, 17–36. https://proceedings.mlr.press/v27/bengio12a.html

Benidis, K., Rangapuram, S. S., Flunkert, V., Wang, Y., Maddix, D., Turkmen, C., Gasthaus, J., Bohlke-Schneider, M., Salinas, D., Stella, L., Aubet, F.-X., Callot, L., & Januschowski, T. (2020, April 21). *Deep Learning for Time Series Forecasting: Tutorial and Literature Survey*. arXiv.Org. https://doi.org/10.1145/3533382

Casolaro, A., Capone, V., Iannuzzo, G., & Camastra, F. (2023). Deep Learning for Time Series Forecasting: Advances and Open Problems. *Information*, *14*(11), Article 11. https://doi.org/10.3390/info14110598

Doshi-Velez, F., & Kim, B. (2017). *Towards A Rigorous Science of Interpretable Machine Learning* (arXiv:1702.08608). arXiv. https://doi.org/10.48550/arXiv.1702.08608

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., & Hutter, F. (2015). Efficient and Robust Automated Machine Learning. *Advances in Neural Information Processing Systems*, *28*. https://papers.nips.cc/paper_files/paper/2015/hash/11d0e6287202fced83f79975ec59a3a6-Abstract.html

Gal, Y., & Ghahramani, Z. (2016). *A Theoretically Grounded Application of Dropout in Recurrent Neural Networks* (arXiv:1512.05287). arXiv. https://doi.org/10.48550/arXiv.1512.05287

Gomez, A. N., Ren, M., Urtasun, R., & Grosse, R. B. (2017). The Reversible Residual Network: Backpropagation Without Storing Activations. *Advances in Neural Information Processing Systems*, *30*. https://proceedings.neurips.cc/paper_files/paper/2017/hash/f9be311e65d81a9ad8150a60844bb94c-Abstract.html

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Howard, A., Inversion, Makridakis, S., & Vangelis. (2020, March 24). *M5 Forecasting—Accuracy*. M5 Forecasting - Accuracy. https://kaggle.com/competitions/m5-forecasting-accuracy

Hyndman, R. J., & Athanasopoulos, G. (2021). *Forecasting: Principles and Practice (3rd ed)*. https://otexts.com/fpp3/

Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, *22*(4), 679–688. https://doi.org/10.1016/j.ijforecast.2006.03.001

Kitaev, N., Kaiser, Ł., & Levskaya, A. (2020). *Reformer: The Efficient Transformer* (arXiv:2001.04451). arXiv. https://doi.org/10.48550/arXiv.2001.04451

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. Proceedings of the IEEE. https://doi.org/10.1109/5.726791

Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.-X., & Yan, X. (2019). Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting. *Advances in Neural Information Processing Systems*, *32*. https://proceedings.neurips.cc/paper/2019/hash/6775a0635c302542da2c32aa19d86be0-Abstract.html

Liu, Y., Wu, H., Wang, J., & Long, M. (2022). Non-stationary Transformers: Exploring the Stationarity in Time Series Forecasting. *Advances in Neural Information Processing Systems*, *35*, 9881–9893.

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2022a). M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*, *38*(4), 1346–1364. https://doi.org/10.1016/j.ijforecast.2021.11.013

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2022b). The M5 competition: Background, organization, and implementation. *International Journal of Forecasting*, *38*(4), 1325–1336. https://doi.org/10.1016/j.ijforecast.2021.07.007

Nie, Y., Nguyen, N. H., Sinthong, P., & Kalagnanam, J. (2023). *A Time Series is Worth 64 Words: Long-term Forecasting with Transformers* (arXiv:2211.14730). arXiv. https://doi.org/10.48550/arXiv.2211.14730

Salinas, D., Bohlke-Schneider, M., Callot, L., Medico, R., & Gasthaus, J. (2019, October 7). *High-Dimensional Multivariate Forecasting with Low-Rank Gaussian Copula Processes*. arXiv.Org. https://arxiv.org/abs/1910.03002v2

Taylor, S. J., & Letham, B. (2018). Forecasting at Scale. *The American Statistician*, *72*(1), 37–45. https://doi.org/10.1080/00031305.2017.1380080

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. ukasz, & Polosukhin, I. (2017). Attention is All you Need. *Advances in Neural Information Processing Systems*, *30*.

https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

Woo, G., Liu, C., Sahoo, D., Kumar, A., & Hoi, S. (2022). *ETSformer: Exponential Smoothing Transformers for Time-series Forecasting* (arXiv:2202.01381). arXiv. https://doi.org/10.48550/arXiv.2202.01381

Wu, H., Xu, J., Wang, J., & Long, M. (2021). Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. *Advances in Neural Information Processing Systems*, *34*, 22419–22430. https://proceedings.neurips.cc/paper/2021/hash/bcc0d400288793e8bdcd7c19a8ac0c2b-Abstract.html

Zhang, Y., & Yan, J. (2022, September 29). *Crossformer: Transformer Utilizing Cross-Dimension Dependency for Multivariate Time Series Forecasting*. The Eleventh International Conference on Learning Representations. https://openreview.net/forum?id=vSVLM2j9eie

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., & Zhang, W. (2021). Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(12), Article 12. https://doi.org/10.1609/aaai.v35i12.17325

Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., & Jin, R. (2022). *FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting* (arXiv:2201.12740). arXiv. http://arxiv.org/abs/2201.12740