mathematics

MDPI

Article

Evaluating the Effectiveness of Time Series Transformers for Demand Forecasting in Retail

José Manuel Oliveira [1,2,*,†] and Patrícia Ramos [2,3,†]

[1] Faculty of Economics, University of Porto, Rua Dr. Roberto Frias, 4200-464 Porto, Portugal
[2] Institute for Systems and Computer Engineering, Technology and Science, Campus da FEUP, Rua Dr. Roberto Frias, 4200-465 Porto, Portugal
[3] CEOS.PP, ISCAP, Polytechnic of Porto, Rua Jaime Lopes Amorim s/n, 4465-004 São Mamede de Infesta, Portugal
* Correspondence: jmo@fep.up.pt
† These authors contributed equally to this work.

Abstract: This study investigates the effectiveness of Transformer-based models for retail demand forecasting. We evaluated vanilla Transformer, Informer, Autoformer, PatchTST, and temporal fusion Transformer (TFT) against traditional baselines like AutoARIMA and AutoETS. Model performance was assessed using mean absolute scaled error (MASE) and weighted quantile loss (WQL). The M5 competition dataset, comprising 30,490 time series from 10 stores, served as the evaluation benchmark. The results demonstrate that Transformer-based models significantly outperform traditional baselines, with Transformer, Informer, and TFT leading the performance metrics. These models achieved MASE improvements of 26% to 29% and WQL reductions of up to 34% compared to the seasonal Naïve method, particularly excelling in short-term forecasts. While Autoformer and PatchTST also surpassed traditional methods, their performance was slightly lower, indicating the potential for further tuning. Additionally, this study highlights a trade-off between model complexity and computational efficiency, with Transformer models, though computationally intensive, offering superior forecasting accuracy compared to the significantly slower traditional models like AutoARIMA. These findings underscore the potential of Transformer-based approaches for enhancing retail demand forecasting, provided the computational demands are managed effectively.

Keywords: Transformer; time series forecasting; quantile forecasting; retail; Informer; Autoformer; PatchTST; TFT

MSC: 68T07

Citation: Oliveira, J.M.; Ramos, P. Evaluating the Effectiveness of Time Series Transformers for Demand Forecasting in Retail. *Mathematics* **2024**, *12*, 2728. https://doi.org/10.3390/math12172728

Academic Editors: Shuo Yu and Feng Xia

Received: 27 July 2024
Revised: 26 August 2024
Accepted: 29 August 2024
Published: 31 August 2024

1. Introduction

The global retail industry, a behemoth estimated at a staggering USD 25 trillion [1], hinges on a fundamental principle: ensuring the expeditious orchestration of product flow from supplier to consumer. This translates to a large-scale supply chain optimization challenge, where even minor revenue gains can significantly impact a retailer's profitability due to high fixed costs and narrow margins. This necessitates the adoption of cutting-edge technological advancements to enhance demand forecasting accuracy [2]. Sales forecasts serve as the cornerstone for various critical business decisions, encompassing workforce scheduling, inventory replenishment, and safety stock calculations [3]. Operational decisions often take place at the product-store level, and typically on a daily or weekly basis. As a result, granular forecasting methods are required to effectively manage a wide range of time series data, which may show varying levels of volatility and intermittency (periods with no sales) [4].

Transformers, a class of machine learning models centered on the self-attention mechanism, have revolutionized various fields since their inception for natural language processing. Their capacity to efficiently process sequential data, coupled with their ability to

exploit parallel computation, has led to state-of-the-art performance across diverse domains, including image recognition, natural language generation, and reinforcement learning.

In the realm of time series analysis, where data unfolds sequentially over time, RNNs (recurrent neural networks) have been the predominant approach [5,6]. While RNNs, such as LSTM (long short-term memory) cells and GRUs (gated recurrent units), have shown success in capturing temporal dependencies, they often grapple with vanishing gradients, computational inefficiency, and limitations in handling long-range patterns [7].

The Transformer architecture offers a compelling alternative by processing input sequences in parallel and attending to all elements simultaneously. This enables the model to effectively capture complex dependencies within time series data. While this approach holds immense potential, the computational demands of self-attention, particularly when dealing with extensive sequences [8], have necessitated the development of optimized architectures [9]. In response, researchers have introduced several variants of the foundational vanilla Transformer including Autoformer, Informer, TFT, and PatchTST, which have demonstrated remarkable performance across a variety of time series forecasting domains.

While these models have been rigorously evaluated on datasets encompassing energy, traffic, economics, weather, and even public health, a notable gap exists in the literature regarding their application to retail time series data. This absence of research precludes a comprehensive understanding of their efficacy and potential limitations in the context of retail forecasting.

This paper offers a systematic evaluation of leading Transformer architectures applied to time series forecasting within the retail domain. By utilizing the widely recognized M5 dataset, we provide:

- A benchmark for Transformer-based retail forecasting: Our study establishes a robust baseline for future research by rigorously evaluating the performance of various Transformer variants on a real-world retail dataset.
- Comparative analysis of point and probabilistic forecasts: We contribute to the understanding of Transformer capabilities by examining their performance in both point and probabilistic forecasting scenarios within the retail context.
- Practical guidelines for Transformer implementation: Our research offers actionable insights and best practices for effectively training Transformers on retail time series data, including techniques to overcome common challenges.
- Advancement of Transformer knowledge in retail: By systematically exploring the potential of Transformers in the retail domain, our work expands the body of knowledge on applying this powerful technique to this critical industry.

The paper is structured as follows. Section 2 provides a comprehensive overview of the evolution of deep learning techniques for time series forecasting, emphasizing the transition from recurrent neural networks to groundbreaking Transformer architecture. The section highlights the Transformer's ability to efficiently capture long-range dependencies and its superior performance compared to RNN-based models. Section 3 delves into the intricacies of the Transformer architecture, explaining its core components and the rationale behind its design. Furthermore, it explores key variants of the Transformer that have emerged in recent years. Section 4 introduces traditional time series forecasting methods, serving as essential baselines for comparison. Section 5 presents a rigorous empirical evaluation of the proposed models on the M5 dataset, including a detailed analysis of performance metrics and hyperparameter tuning. Finally, Section 6 summarizes the key findings, contributions, and potential avenues for future research.

2. Related Work

In the domain of machine learning, deep learning algorithms have established themselves as the leading methodologies, achieving notable success across multiple computer science fields, such as computer vision, natural language processing, and speech recognition. Lately, there has been a surge in the use of these algorithms for time series forecasting, largely due to their ability to capture complex, non-linear relationships within

the data [10–12]. Among deep learning algorithms, recurrent neural networks (RNNs) stand out as particularly noteworthy [13,14]. However, traditional RNNs face challenges in training stability, which hinders their ability to capture long-range dependencies. Similar to most neural networks, RNNs rely on gradient descent for training, with backpropagation through time applied to calculate the gradient of the loss function concerning the network's weights. In deep networks, backpropagation can lead to problems such as vanishing or exploding gradients. As the error signal propagates backward, certain components may become negligibly small, resulting in minimal weight updates, or excessively large, causing training instability. To address these gradient-related issues, various techniques have been developed over time. Long short-term memory (LSTM) cells and gated recurrent units (GRUs) have proven particularly effective [15]. These gated mechanisms regulate the flow of information through the network and help mitigate gradient instability. However, even with stable training, RNNs can still struggle to learn relationships between distant elements in a sequence. The limited memory of RNNs is a significant drawback, prompting the introduction of attention mechanisms and Transformers in deep learning. Furthermore, RNNs typically do not take full advantage of parallel processing capabilities offered by modern hardware accelerators such as graphical processing units (GPUs) and tensor processing units (TPUs) [16].

Transformers leverage a unique architecture enabling highly parallelized computation for sequential data [17,18]. This translates to significant speed improvements without bloating the network's complexity. This efficiency stems from their core design, allowing them to exploit the parallel processing prowess of GPUs and TPUs. Due to the attention-based mechanisms, Transformers can efficiently relate information from all elements within a sequence simultaneously, avoiding the vanishing gradient issues that plague RNNs and their derivatives. This architectural advantage has led to remarkable advancements in long-term and multivariate time series forecasting. However, the computational demands and memory requirements of the self-attention component pose challenges for modeling extensive sequences. Researchers have addressed this hurdle by proposing various performance optimization techniques specifically tailored for time series tasks.

Informer's architecture [19] incorporates three significant innovations to enhance efficiency and performance. The first is the ProbSparse self-attention mechanism, which reduces the time complexity and memory usage of self-attention from $\mathcal{O}(L^2)$ to $\mathcal{O}(L \log L)$. In this context, $\mathcal{O}$, known as Big O notation, describes the upper bound of the algorithm's complexity, indicating how the computational resources required (time and memory) grow with the length of the series, denoted by L. The ProbSparse mechanism achieves this reduction by focusing on the most informative query-key pairs, thereby maintaining high dependency alignment with significantly reduced computational cost. The second innovation is the self-attention distilling technique, which prioritizes dominant attention scores across layers. This technique progressively reduces input length and halves the cascading layer input, efficiently handling extremely long sequences. As a result, the space complexity is reduced to $\mathcal{O}((2-\epsilon)L \log L)$. The third innovation is the generative-style decoder. Unlike traditional methods that predict step-by-step, this decoder predicts the entire sequence in a single forward operation. This approach dramatically improves inference speed and mitigates the accumulation of prediction errors.

Autoformer [20] is an advanced Transformer architecture designed for long-term time series forecasting, featuring two key innovations: a decomposition architecture and an autocorrelation mechanism. The decomposition architecture progressively separates time series data into trend and seasonal components, facilitating the extraction of long-term patterns. The autocorrelation mechanism, replacing traditional self-attention, efficiently identifies periodic dependencies, aggregating similar sub-series to improve computational efficiency and prediction accuracy. This structure, with a complexity of $\mathcal{O}(L \log L)$, allows Autoformer to achieve superior performance in various forecasting tasks by effectively handling complex temporal patterns.

The temporal fusion Transformer (TFT) architecture [21] is designed to address the complexities of multi-horizon time series forecasting by integrating several specialized components. It employs static covariate encoders to generate context vectors from static features, which are utilized across the network. Gating mechanisms and sample-dependent variable selections dynamically filter irrelevant inputs, ensuring focus on pertinent data. A sequence-to-sequence layer processes known and observed inputs locally, capturing short-term dependencies, while a temporal self-attention decoder identifies long-term dependencies by weighing the importance of different time steps. This combination allows TFT to handle diverse input types and provide interpretable insights into temporal dynamics, significantly enhancing forecasting performance and interpretability.

Patch time series Transformer (PatchTST) [22] is a Transformer-based model designed specifically for multivariate time series forecasting, incorporating two primary innovations: segmentation of time series into subseries-level patches and a channel-independent processing framework. Each univariate time series is independently normalized and segmented into patches, which serve as input tokens for the Transformer. These patches are projected into a higher-dimensional space with positional encodings to preserve temporal order. The Transformer encoder, consisting of multi-head attention mechanisms, normalization layers, and feed-forward networks, processes these tokens to generate latent representations. The model then concatenates these representations and passes them through a linear head to produce the forecasted values. This architecture reduces computational complexity by decreasing the number of input tokens and enhances the model's ability to capture long-term dependencies, resulting in superior forecasting accuracy and efficiency compared to traditional Transformer-based models.

The Pyraformer, FEDformer, and non-stationary Transformer are examples of advanced Transformer variants designed for time series forecasting, which are less widely known and not as readily accessible through standard repositories [23–25] compared to more established models like the vanilla Transformer, PatchTST, TFT, Autoformer, and Informer. The Pyraformer architecture [26] introduces a novel pyramidal attention mechanism designed to reduce the computational complexity inherent in modeling long-range dependencies in time series data. It hierarchically aggregates information through multiple layers of attention, effectively capturing both local and global temporal patterns. FEDformer [27], or the frequency-enhanced decomposed Transformer, integrates frequency domain decomposition into the Transformer architecture to enhance long-term time series forecasting. It decomposes time series data into trend and seasonal components using the Fourier transform and processes these components separately using dedicated Transformer layers. The non-stationary Transformer [28] addresses the challenges of non-stationarity in time series data by incorporating a learnable transformation module that adapts the input data to a stationary representation. This transformation is coupled with a standard Transformer architecture, which then processes the now-stationary data for forecasting.

3. Time Series Transformer Models

Introduced by Google in 2017 [29], the Transformer architecture leverages attention mechanisms to excel at processing sequential data. Initially conceived for natural language tasks like machine translation, where it transforms input sequences from one language into output sequences of another, the Transformer's capabilities extend beyond text. By framing time series data as sequential patterns, analogous to sentences in different languages, we can apply the Transformer to tackle the challenge of multi-step time series forecasting. In essence, the Transformer enables us to map historical time series segments to future value predictions.

In this section, we start with an explanation of the core functionalities within the Transformer architecture, as introduced by [29] to tackle the intricate task of neural machine translation. We will then delve into a rigorous exploration of the operations executed within each Transformer component, along with the underlying rationale for these operations. The Transformer architecture has witnessed prolific innovation, with numerous variations emerging in recent years. While many of these modifications have been proposed to

enhance Transformer performance, our focus will be on the most prevalent and well-established variants.

3.1. Vanilla Transformer

Let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_L] \in \mathbb{R}^{L \times d}$ denote a multivariate time series where L is the series's length and each $\mathbf{x}_t \in \mathbb{R}^d$ is a d-dimensional vector depicting the values of the d variables of $\mathbf{X}$ at time t. The time series forecasting task is to predict future values of the target variable at time $L+1, \ldots, L+H$ where H is the forecast horizon.

Transformers, due to their attention mechanisms, treat input data as unordered sets rather than sequential structures. Consequently, explicit encoding of temporal relationships is essential for time series forecasting. This is typically accomplished by integrating positional information into the input embeddings. A common strategy involves adding a positional embedding to the projected input representation (see Figure 1). For this purpose, one-dimensional convolutional layers can be employed to extract a higher D-dimensional feature vector for each data point. Positional embeddings can be learned parameters or fixed functions, such as sinusoidal encodings, or temporal-based alternatives. While distinct embeddings for the encoder and decoder are feasible, sharing a single embedding is also a valid approach when input and output data share the same domain.

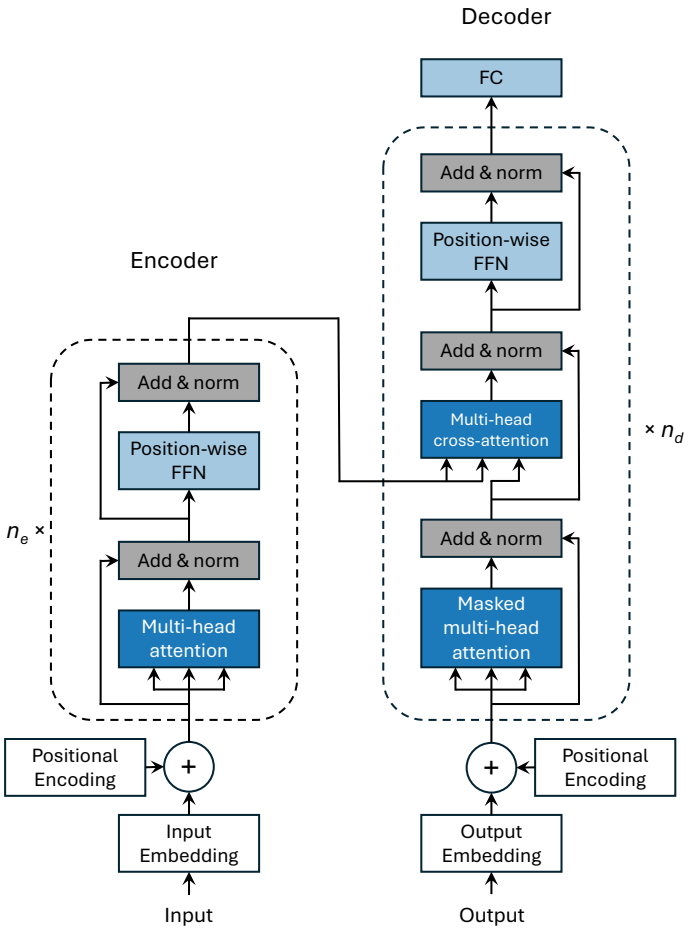

Figure 1. Transformer model architecture. The encoder component, positioned on the left, processes the input generating a latent representation. The decoder component, positioned on the right, leverages this representation to produce the output in an autoregressive manner. This means that the decoder iteratively generates output elements, using previously produced elements as additional input for subsequent predictions.

The encoder and decoder components are built from n_e and n_d successive layers, respectively, as illustrated in Figure 1. Data flow sequentially through these layers, with each

layer's output serving as input for the subsequent one. Encoder layers comprise two sub-layers: a self-attention mechanism that establishes relationships between input elements and a position-wise feed-forward neural network (FFN) acting as a nonlinear transformation. To facilitate gradient flow during training, residual connections are employed between each sub-layer and its input, and layer normalization is applied to standardize the output sequence [30]. Decoder layers adopt a similar architecture to encoder layers but include an additional sub-layer. The initial sub-layer is a masked self-attention mechanism, preventing the model from accessing future data points while processing the current one. This output is then integrated with the encoder's final hidden representation through a cross-attention layer, followed by another position-wise feed-forward neural network. Residual connections and layer normalization are also incorporated into decoder sub-layers. The final decoder layer feeds into a prediction layer to generate the desired output.

Core computational units within Transformer models are attention mechanisms. These mechanisms enable the model to selectively concentrate on specific input segments based on the processed information. Among various attention formulations, Transformers employ scaled dot-product attention, which bears a strong resemblance to multiplicative attention. Attention computations involve three core components: queries $\mathbf{Q}$, keys $\mathbf{K}$, and values $\mathbf{V}$. By multiplying the input matrix $\mathbf{X} \in \mathbb{R}^{L \times D}$ with learnable query, key, and value weight matrices, we obtain $\mathbf{Q} \in \mathbb{R}^{L \times D_k}$, $\mathbf{K} \in \mathbb{R}^{L \times D_k}$, and $\mathbf{V} \in \mathbb{R}^{L \times D_v}$ respectively:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}^Q, \quad \mathbf{K} = \mathbf{X}\mathbf{W}^K, \quad \mathbf{V} = \mathbf{X}\mathbf{W}^V. \tag{1}$$

Leveraging these matrices, we compute query-key comparisons through the multiplication of $\mathbf{Q}$ and $\mathbf{K}^T$, followed by scaling, softmax application, and multiplication with $\mathbf{V}$, resulting in an $L \times D$ matrix. To mitigate numerical instabilities and gradient vanishing during training, the dot product is scaled by dividing it by the square root of the key vector dimensionality D_k. The self-attention output, a matrix where each row represents the output vector for a corresponding query, is calculated as follows:

$$\text{SelfAttention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{D_k}}\right)\mathbf{V}. \tag{2}$$

Transformers utilize attention in two primary modes: self-attention and cross-attention. In self-attention, queries, keys, and values are all derived from the same input sequence, allowing the model to attend to different parts of the input itself. Conversely, in cross-attention, queries originate from the preceding decoder sublayer, while key-value pairs stem from the encoder's hidden representation, facilitating interaction between the input and output sequences.

A single self-attention mechanism encounters challenges in capturing the diverse parallel relationships inherent in its inputs. To overcome this limitation, Transformers employ multi-head self-attention layers [29]. These layers consist of multiple parallel self-attention heads, each equipped with its own parameter set. This arrangement enables each head to independently learn distinct facets of input interconnections at the same abstraction level. Concretely, each head, denoted by index i, is furnished with separate query, key, and value weight matrices: $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$, and $\mathbf{W}_i^V$. These matrices project inputs into distinct query, key, and value embeddings for each head while preserving the core self-attention computations. Analogous to standard self-attention, the input and output dimensions remain as D, while key and query embeddings adopt dimensionality D_k, and value embeddings utilize D_v. Consequently, for each head i, we have weight matrices $\mathbf{W}_i^Q \in \mathbb{R}^{D \times D_k}$, $\mathbf{W}_i^K \in \mathbb{R}^{D \times D_k}$, and $\mathbf{W}_i^V \in \mathbb{R}^{D \times D_v}$. Multiplying these matrices by the input matrix $\mathbf{X} \in \mathbb{R}^{L \times D}$ generates $\mathbf{Q}_i \in \mathbb{R}^{L \times D_k}$, $\mathbf{K}_i \in \mathbb{R}^{L \times D_k}$, and $\mathbf{V}_i \in \mathbb{R}^{L \times D_v}$ The output of each of the h heads has dimensions $L \times D_v$. To integrate these outputs, they are concatenated into a single matrix of size $L \times hD_v$. Finally, a linear projection $\mathbf{W}^O \in \mathbb{R}^{hD_v \times D}$ reshapes this matrix to the original output dimension D for each data point. Multiplying the

concatenated matrix by $\mathbf{W}^O$ produces the self-attention output with dimensions $L \times D$, suitable for subsequent residual connections and layer normalization, as follows:

$$\mathbf{Q}_i = \mathbf{X}\mathbf{W}_i^Q,\ \mathbf{K}_i = \mathbf{X}\mathbf{W}_i^K,\ \mathbf{V}_i = \mathbf{X}\mathbf{W}_i^V, \quad \mathbf{h}_i = \text{SelfAttention}(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i),\ i = 1, \ldots, h\,, \quad (3)$$

$$\text{MultiHeadAttention}(\mathbf{X}) = \text{Concatenate}(\mathbf{h}_1, \ldots, \mathbf{h}_h)\mathbf{W}^O\,. \quad (4)$$

Transformer models exhibit three primary limitations. Firstly, they demonstrate a deficiency in capturing local dependencies. The scaled dot-product attention mechanism (Equation (2)) is insensitive to proximal patterns, rendering the model vulnerable to anomalies in time series forecasting. Secondly, Transformers suffer from a computational bottleneck characterized by quadratic space complexity, scaling with the square of the series's length $\mathcal{O}(L^2)$. This quadratic growth in memory requirements and computational cost restricts their applicability to extended time series. To address these shortcomings, various Transformer variants have been proposed, as detailed in the subsequent section.

3.2. Adaptive Transformer Architectures for Time Series Forecasting

In the realm of long sequence time series forecasting, traditional Transformer models face significant challenges due to their quadratic time complexity, high memory usage, and inherent limitations in handling long sequences. The Informer architecture [19] addresses these issues with three core innovations: the ProbSparse self-attention mechanism, self-attention distilling, and a generative style decoder. Firstly, the ProbSparse self-attention mechanism is designed to enhance computational efficiency. Traditional self-attention mechanisms in Transformers have a time complexity of $\mathcal{O}(L^2)$, which becomes infeasible for long sequences due to the exponential growth in computational and memory requirements. The ProbSparse self-attention mechanism mitigates this by focusing only on the most critical query-key pairs, thereby reducing the time complexity and memory usage to $\mathcal{O}(L \log L)$. This selective attention is based on a sparsity measurement that identifies and prioritizes dominant attention scores, allowing the model to maintain performance while significantly reducing computational overhead. Secondly, Informer introduces self-attention distilling, a technique that further enhances efficiency by iteratively compressing the input sequence length across layers. This method highlights the dominant attention scores, effectively reducing the amount of data processed at each layer and subsequently lowering the space complexity to $\mathcal{O}((2-\epsilon)L \log L)$. By halving the input sequence length in cascading layers, self-attention distilling ensures that the model can handle extremely long sequences without a prohibitive increase in memory usage. Lastly, the generative style decoder distinguishes Informer from traditional Transformers by predicting the entire output sequence in a single forward operation rather than through a step-by-step decoding process. This approach not only accelerates the inference process but also reduces the cumulative error that typically accrues in sequential prediction models. By treating the prediction of long time series sequences as a single generative task, Informer achieves faster and more accurate long-sequence forecasts. The Informer architecture was validated through extensive experiments using several datasets including electricity transformer temperature (ETT), electricity consumption load (ECL), and Weather. These experiments demonstrated its superior performance compared to existing methods.

Autoformer [20] is a sophisticated Transformer architecture specifically designed to address the challenges of long-term time series forecasting by incorporating two key innovations: a decomposition architecture and an autocorrelation mechanism. The architecture follows a standard encoder–decoder structure but introduces a series decomposition block as a fundamental operation. This block progressively separates time series into trend-cyclical and seasonal components, leveraging moving average techniques to isolate long-term trends and cyclical patterns from intermediate predictions. This progressive decomposition enables Autoformer to handle complex temporal patterns more effectively than traditional models. The autocorrelation mechanism replaces the conventional self-attention mechanism, providing a more efficient and accurate method for dependency

discovery and information aggregation. By utilizing the inherent periodicity of time series data, the autocorrelation mechanism identifies and aggregates similar sub-series across different periods, significantly improving both computational efficiency and information utilization. This mechanism achieves a computational complexity of $\mathcal{O}(L \log L)$, making it more suitable for long-term forecasting tasks compared to the quadratic complexity of standard self-attention mechanisms. In the encoder, the model focuses on capturing seasonal patterns by eliminating long-term trends through decomposition blocks. The encoder's output, which contains seasonal information, is then used in the decoder to refine predictions. The decoder, structured to progressively accumulate and refine trend-cyclical components, combines the seasonal insights from the encoder with its own decomposition process to enhance forecast accuracy. Extensive experimentation on multiple benchmarks, including applications in energy, traffic, economics, weather, and disease forecasting, demonstrates Autoformer's superior performance. It achieves state-of-the-art accuracy, showing a 38% relative improvement over previous models, thereby establishing its effectiveness in handling long-term time series forecasting tasks.

The temporal fusion Transformer (TFT) architecture [21] is a sophisticated model designed for high-performance multi-horizon forecasting, integrating various components to effectively handle both static and time-varying inputs. The TFT is particularly notable for its ability to provide interpretability alongside its forecasting capabilities, making it a valuable tool in time series analysis. At its core, the TFT architecture is built around several key components that work together to process different types of input data. The TFT employs gated residual networks (GRNs) to manage the flow of information through the model. These gating mechanisms allow the architecture to adaptively skip over unnecessary components, providing flexibility in processing. This is particularly useful in scenarios where the relevance of certain inputs may vary, enabling the model to apply complex non-linear transformations only when needed. To enhance the model's interpretability and efficiency, TFT incorporates variable selection networks that identify and select the most relevant input variables at each time step. This feature ensures that the model focuses on the most salient information, reducing noise from irrelevant inputs and improving overall forecasting accuracy. The architecture includes specialized encoders for static covariates, which are features that do not change over time. These encoders create context vectors that condition the temporal dynamics of the model, allowing it to integrate static information effectively into the forecasting process. TFT utilizes a combination of sequence-to-sequence layers and multi-head attention mechanisms to capture both short-term and long-term temporal relationships within the data. The sequence-to-sequence layer processes known and observed inputs locally, while the attention mechanism enables the model to learn dependencies across different time steps, enhancing its ability to recognize patterns and trends. To provide a comprehensive view of the predicted outcomes, TFT incorporates quantile regression techniques. This allows the model to generate prediction intervals, offering insights into the range of likely target values at each prediction horizon. This feature is particularly beneficial in applications where understanding uncertainty is crucial. The TFT architecture is designed to handle a wide range of forecasting tasks, from simple datasets with known inputs to complex scenarios involving multiple types of data. By aligning its structure with the intricacies of multi-horizon forecasting, TFT achieves state-of-the-art performance across various datasets. Moreover, the interpretability of the TFT model is a significant advantage. It allows users to analyze important variables for specific prediction problems, visualize persistent temporal relationships (such as seasonality), and identify significant regime changes in the data. This interpretability is facilitated by the attention mechanisms, which provide insights into how different inputs contribute to the model's predictions.

Patch time series Transformer (PatchTST) [22], a novel architecture designed for multivariate time series forecasting and self-supervised representation learning, leverages the strengths of Transformers with two key innovations: segmentation of time series into subseries-level patches and a channel-independent processing approach. This model ad-

dresses several challenges associated with traditional Transformer models, particularly in handling long-term dependencies and computational efficiency. In PatchTST, time series data are divided into subseries-level patches, which serve as input tokens for the Transformer. This segmentation has multiple benefits. By aggregating time steps into patches, the model retains local semantic information, akin to how words are handled in natural language processing. The patching mechanism significantly reduces the number of input tokens N, which in turn reduces the computation and memory usage of attention maps from $\mathcal{O}(N^2)$ to $\mathcal{O}((L/S)^2)$, where L is the original sequence length and S is the stride length. This reduction in token length allows the model to attend to longer historical data without an increase in computational burden, enhancing the model's ability to capture long-term dependencies. PatchTST processes each univariate time series independently through a shared Transformer backbone. This channel-independent approach contrasts with channel-mixing strategies, where information from multiple channels is combined. The advantages of this method include (1) scalability: each channel's data are processed separately, making the model scalable and efficient, especially for high-dimensional multivariate time series; and (2) modularity: by treating each channel independently, the model simplifies the integration of new data channels and facilitates parallel processing. PatchTST's architecture is built on a vanilla Transformer encoder. The process involves instance Normalization and Patching, i.e., each univariate series undergoes instance normalization and is segmented into patches. These patches are projected into a higher-dimensional space and embedded with positional encodings to maintain temporal order. The patches are processed by a multi-head attention mechanism, followed by normalization, feed-forward layers, and residual connections. This results in a latent representation of the time series. The latent representations are flattened and passed through a linear head to generate the forecasted values. Experiments demonstrate PatchTST's superior performance in both supervised and self-supervised settings. When applied to datasets like traffic, electricity, and weather, PatchTST significantly reduced the mean squared error (MSE) compared to state-of-the-art models. For instance, with a look-back window $L = 336$, the model achieved an MSE of 0.367 using patching, compared to 0.518 without patching [22]. Additionally, self-supervised pretraining further enhanced the model's accuracy, achieving the best MSE of 0.349. For more in-depth information about each Transformer variant, please refer to the original papers where the architectures are introduced.

4. Baselines

To objectively assess the performance of forecasting models, researchers and practitioners often employ baseline models as a reference point [31]. These baselines serve as a standard against which the accuracy and effectiveness of novel forecasting methods can be compared. By establishing a benchmark, it becomes possible to quantify the improvement offered by a new model and to discern its added value in specific forecasting scenarios. In essence, benchmarking provides a rigorous framework for evaluating the relative performance of different models, enabling informed decisions about model selection and application.

Two of the most widely used benchmark models for time series forecasting are exponential smoothing (ETS) and autoregressive integrated moving average (ARIMA) [32]. These models have been extensively studied and applied in various domains, providing a solid foundation for comparison. By establishing these models as a baseline, researchers can gauge the effectiveness of more sophisticated techniques and identify areas for potential improvement. Similarly, to establish a baseline for evaluating the performance of time series forecasting models, the seasonal Naïve method is often employed [33]. This simple yet effective approach leverages the recurring patterns present in seasonal data to generate forecasts. By replicating the values from the same period in the previous season, the seasonal Naïve method provides a straightforward and computationally efficient estimate of future values. While its simplicity limits its ability to capture complex patterns, it serves as a valuable benchmark for assessing the added value of more sophisticated models. The

following sections will provide an overview of ARIMA and ETS models, delving into their underlying principles and methodologies. The Naïve and seasonal Naïve forecasting methods will also be discussed.

4.1. ARIMA Models

The seasonal ARIMA model, represented as ARIMA$(p,d,q) \times (P,D,Q)_m$, is a model specifically designed for stationary time series. It can be formulated as follows:

$$\phi_p(B)\Phi_P(B^m)(1-B)^d(1-B^m)^D\eta_t = c + \theta_q(B)\Theta_Q(B^m)\varepsilon_t\,, \tag{5}$$

where the regular autoregressive and moving average polynomials, $\phi_p(B)$ and $\theta_q(B)$, are of orders p and q respectively, while the seasonal autoregressive and moving average polynomials, $\Phi_P(B^m)$ and $\Theta_Q(B^m)$, are of orders P and Q respectively. The terms are defined as follows:

$$\begin{aligned} \phi_p(B) &= 1 - \phi_1 B - \cdots - \phi_p B^p\,, & \Phi_P(B^m) &= 1 - \Phi_1 B^m - \cdots - \Phi_P B^{Pm}\,, \\ \theta_q(B) &= 1 + \theta_1 B + \cdots + \theta_q B^q\,, & \Theta_Q(B^m) &= 1 + \Theta_1 B^m + \cdots + \Theta_Q B^{Qm}\,, \end{aligned}$$

where η_t represents the target time series, m is the seasonal period, and D and d denote the degrees of seasonal and ordinary differencing respectively, and B is the backward shift operator. Here, $c = \mu(1-\phi_1-\cdots-\phi_p)(1-\Phi_1-\cdots-\Phi_P)$ with μ being the mean of $(1-B)^d(1-B^m)^D\eta_t$, and ε_t a white noise series with zero mean and constant variance. For the time series to be stationary and invertible, the zeros of the polynomials $\phi_p(B)$, $\Phi_P(B^m)$, $\theta_q(B)$, and $\Theta_Q(B^m)$ must all lie outside the unit circle. Non-stationary time series can be transformed into stationary series by applying logarithmic transformations to stabilize the variance and by taking appropriate degrees of differencing to stabilize the mean. Once the values for p, q, P, and Q are specified, the model parameters $c, \phi_1, \ldots, \phi_p, \theta_1, \ldots, \theta_q, \Phi_1, \ldots, \Phi_P, \Theta_1, \ldots, \Theta_Q$ can be estimated by maximizing the log-likelihood function. The values of p, q, P, and Q can be determined using Akaike's Information Criterion (AIC), which balances the log-likelihood with a regularization term to penalize model complexity and prevent overfitting. For implementing ARIMA models, we utilized the `AutoARIMA` function from the `StatsForecast` Python library [34], which mirrors Hyndman's [35] `auto.arima` function in the `forecast` package of the R programming language.

4.2. Exponential Smoothing Models

Exponential smoothing methodologies are characterized by the interplay of observation and state equations. The former delineates the connection between the time series and its constituent elements—level, trend, and seasonality—while the latter elucidates how these components evolve temporally [36,37]. These components can interact additively (A) or multiplicatively (M), with the potential for additive or multiplicative damped trends (A_d, M_d). Error terms can be incorporated additively or multiplicatively into each model. The smoothing parameter governs the extent to which the error process modifies each component. For in-depth exploration, the reader is directed to [38,39]. Table 1 presents the state-space formulation for these models, where y_t represents the time series observation at period t, l_t denotes the local level, b_t signifies the local trend, s_t encapsulates local seasonality, and m corresponds to the seasonal period. The smoothing parameters are denoted by α, β, γ, and ϕ, and the error term ε_t is assumed to be normally and independently distributed with zero mean and variance σ^2, expressed as $\varepsilon_t \sim N(0, \sigma^2)$. The Python library `StatsForecast`, specifically the `AutoETS` function, was employed to implement these models [34]. This function mirrors the `ets` function within the R forecast package by [37].

Table 1. State-space formulation for exponential smoothing models with additive and multiplicative error.

Additive Error Models

Trend Component	Seasonal Component: N	Seasonal Component: A	Seasonal Component: M
N	$y_t = l_{t-1} + \varepsilon_t$ $l_t = l_{t-1} + \alpha\varepsilon_t$	$y_t = l_{t-1} + s_{t-m} + \varepsilon_t$ $l_t = l_{t-1} + \alpha\varepsilon_t$ $s_t = s_{t-m} + \gamma\varepsilon_t$	$y_t = l_{t-1}s_{t-m} + \varepsilon_t$ $l_t = l_{t-1} + \alpha\varepsilon_t / s_{t-m}$ $s_t = s_{t-m} + \gamma\varepsilon_t / l_{t-1}$
A	$y_t = l_{t-1} + b_{t-1} + \varepsilon_t$ $l_t = l_{t-1} + b_{t-1} + \alpha\varepsilon_t$ $b_t = b_{t-1} + \beta\varepsilon_t$	$y_t = l_{t-1} + b_{t-1} + s_{t-m} + \varepsilon_t$ $l_t = l_{t-1} + b_{t-1} + \alpha\varepsilon_t$ $b_t = b_{t-1} + \beta\varepsilon_t$ $s_t = s_{t-m} + \gamma\varepsilon_t$	$y_t = (l_{t-1} + b_{t-1})s_{t-m} + \varepsilon_t$ $l_t = l_{t-1} + b_{t-1} + \alpha\varepsilon_t / s_{t-m}$ $b_t = b_{t-1} + \beta\varepsilon_t / s_{t-m}$ $s_t = s_{t-m} + \gamma\varepsilon_t / (l_{t-1} + b_{t-1})$
$\mathbf{A_d}$	$y_t = l_{t-1} + \phi b_{t-1} + \varepsilon_t$ $l_t = l_{t-1} + \phi b_{t-1} + \alpha\varepsilon_t$ $b_t = \phi b_{t-1} + \beta\varepsilon_t$	$y_t = l_{t-1} + \phi b_{t-1} + s_{t-m} + \varepsilon_t$ $l_t = l_{t-1} + \phi b_{t-1} + \alpha\varepsilon_t$ $b_t = \phi b_{t-1} + \beta\varepsilon_t$ $s_t = s_{t-m} + \gamma\varepsilon_t$	$y_t = (l_{t-1} + \phi b_{t-1})s_{t-m} + \varepsilon_t$ $l_t = l_{t-1} + \phi b_{t-1} + \alpha\varepsilon_t / s_{t-m}$ $b_t = \phi b_{t-1} + \beta\varepsilon_t / s_{t-m}$ $s_t = s_{t-m} + \gamma\varepsilon_t / (l_{t-1} + \phi b_{t-1})$

Multiplicative Error Models

Trend Component	Seasonal Component: N	Seasonal Component: A	Seasonal Component: M
N	$y_t = l_{t-1}(1 + \varepsilon_t)$ $l_t = l_{t-1}(1 + \alpha\varepsilon_t)$	$y_t = (l_{t-1} + s_{t-m})(1 + \varepsilon_t)$ $l_t = l_{t-1} + \alpha(l_{t-1} + s_{t-m})\varepsilon_t$ $s_t = s_{t-m} + \gamma(l_{t-1} + s_{t-m})\varepsilon_t$	$y_t = l_{t-1}s_{t-m}(1 + \varepsilon_t)$ $l_t = l_{t-1}(1 + \alpha\varepsilon_t)$ $s_t = s_{t-m}(1 + \gamma\varepsilon_t)$
A	$y_t = (l_{t-1} + b_{t-1})(1 + \varepsilon_t)$ $l_t = (l_{t-1} + b_{t-1})(1 + \alpha\varepsilon_t)$ $b_t = b_{t-1} + \beta(l_{t-1} + b_{t-1})\varepsilon_t$	$y_t = (l_{t-1} + b_{t-1} + s_{t-m})(1 + \varepsilon_t)$ $l_t = l_{t-1} + b_{t-1} + \alpha(l_{t-1} + b_{t-1} + s_{t-m})\varepsilon_t$ $b_t = b_{t-1} + \beta(l_{t-1} + b_{t-1} + s_{t-m})\varepsilon_t$ $s_t = s_{t-m} + \gamma(l_{t-1} + b_{t-1} + s_{t-m})\varepsilon_t$	$y_t = (l_{t-1} + b_{t-1})s_{t-m}(1 + \varepsilon_t)$ $l_t = (l_{t-1} + b_{t-1})(1 + \alpha\varepsilon_t)$ $b_t = b_{t-1} + \beta(l_{t-1} + b_{t-1})\varepsilon_t$ $s_t = s_{t-m}(1 + \gamma\varepsilon_t)$
$\mathbf{A_d}$	$y_t = (l_{t-1} + \phi b_{t-1})(1 + \varepsilon_t)$ $l_t = (l_{t-1} + \phi b_{t-1})(1 + \alpha\varepsilon_t)$ $b_t = \phi b_{t-1} + \beta(l_{t-1} + \phi b_{t-1})\varepsilon_t$	$y_t = (l_{t-1} + \phi b_{t-1} + s_{t-m})(1 + \varepsilon_t)$ $l_t = l_{t-1} + \phi b_{t-1} + \alpha(l_{t-1} + \phi b_{t-1} + s_{t-m})\varepsilon_t$ $b_t = \phi b_{t-1} + \beta(l_{t-1} + \phi b_{t-1} + s_{t-m})\varepsilon_t$ $s_t = s_{t-m} + \gamma(l_{t-1} + \phi b_{t-1} + s_{t-m})\varepsilon_t$	$y_t = (l_{t-1} + \phi b_{t-1})s_{t-m}(1 + \varepsilon_t)$ $l_t = (l_{t-1} + \phi b_{t-1})(1 + \alpha\varepsilon_t)$ $b_t = \phi b_{t-1} + \beta(l_{t-1} + \phi b_{t-1})\varepsilon_t$ $s_t = s_{t-m}(1 + \gamma\varepsilon_t)$

4.3. Naïve

The Naïve method is a basic forecasting technique that assumes the next observation in a time series will be equal to the previous observation. In other words, it predicts no change from the most recent data point. Mathematically, this can be expressed as follows:

$$\hat{y}_t = y_{t-1}\,, \tag{6}$$

where $\hat{y}_t$ is the forecasted value at time t, and y_{t-1} is the observed value at time $t-1$. While extremely simple, the Naïve method can serve as a benchmark for evaluating the performance of more complex forecasting models. It is particularly useful when there is no clear trend, seasonal pattern, or other discernible structure in the data.

4.4. Seasonal Naïve

The seasonal Naïve method is also a straightforward approach to time series forecasting that leverages the seasonal patterns inherent in data. This model posits that the future value of a series can be accurately predicted by the most recent observation from the corresponding period in the past. Mathematically, this can be expressed as follows:

$$\hat{y}_t = y_{t-m}\,, \tag{7}$$

where $\hat{y}_t$ is the forecasted value at time t, y_{t-m} is the observed value m periods prior to t, and m represents the length of the seasonal cycle (e.g., 12 for monthly data with annual seasonality). In essence, the seasonal Naïve model assumes that seasonal patterns persist consistently over time, and therefore, the best prediction for a future value is the equivalent value from the previous occurrence of the same season. While simplistic, this method can also serve as a valuable baseline for comparison with more complex forecasting techniques.

5. Empirical Evaluation

5.1. Dataset

To establish the robustness and generalizability of research findings, it is imperative that they can be independently verified and contrasted with related work. Consequently, this study leverages the widely recognized and publicly accessible M5 competition dataset [40], providing a well-established foundation for our investigation. The M5 competition constituted a rigorous benchmark for forecasting methodologies, demanding the generation of both accurate point estimates and probabilistic intervals for hierarchical time series data. The competition's focus on Walmart, the undisputed global leader in retail revenue, rendered it a particularly challenging and high-stakes endeavor.

The M5 dataset encompasses a hierarchical structure composed of 3049 distinct products classified into three primary categories: Foods, Hobbies, and Household goods. These categories are further subdivided into seven product departments (Foods1, Foods2, Foods3, Hobbies1, Hobbies2, Household1, and Household2). Sales data for these items were collected from ten retail stores distributed across three states: California (CA), Texas (TX), and Wisconsin (WI). California has four stores (CA1, CA2, CA3, and CA4), Texas has three stores (TX1 and TX2), and Wisconsin also has three stores (WI1, WI2, and WI3). The dataset spans approximately 5.4 years, capturing daily sales records from 29 January 2011 to 19 June 2016, resulting in a total of 1969 daily observations.

5.2. Forecasting Design

In this study, we conducted a comprehensive evaluation of five Transformer-based models: the vanilla Transformer, temporal fusion Transformer (TFT), Informer, patch time series Transformer (PatchTST), and Autoformer. These models were assessed using the extensive M5 competition dataset, which comprises 30,490 time series representing the sales for 3049 products across ten distinct retail stores.

To accurately evaluate a model's ability to forecast unseen time series data, it is essential to set aside a portion of the dataset as test data that remains excluded from the training process. This is typically achieved through a train-test split. However, a more precise estimation of the model's performance can be obtained via time series cross-validation (evaluation on a rolling forecasting origin) [38]. This approach evaluates the model's generalization ability across multiple forecast horizons within the same time series, offering a more reliable measure of its predictive accuracy. In this study, we adhered to the framework of the M5 competition, utilizing a forecast horizon of 28 days. To thoroughly evaluate the forecast accuracy, we implemented a cross-validation strategy using the last three windows, each spanning 28 days (please see Figure 2). Specifically, Window 1 encompasses the period from 28 March 2016 to 24 April 2016; Window 2 spans from 25 April 2016 to 22 May 2016; and Window 3 covers 23 May 2016, to 19 June 2016. This methodology ensures a rigorous and comprehensive assessment of the model's forecasting capabilities, offering a robust evaluation of its performance on unseen data.

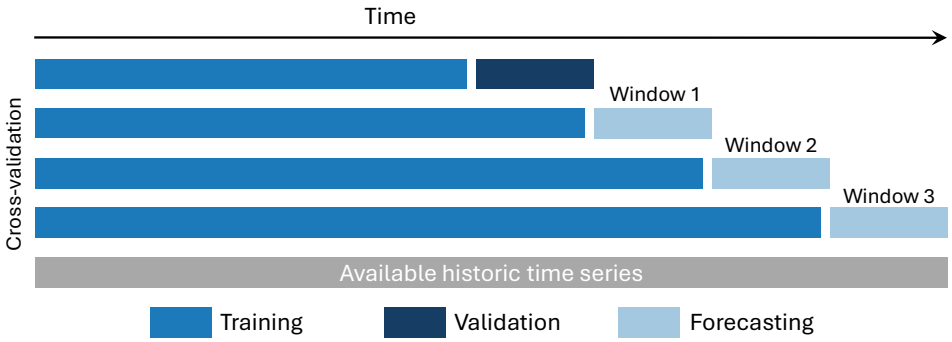

Figure 2. Model training using a validation set split and model evaluation using cross-validation.

5.3. Point and Probabilistic Forecasting

While point forecasts, providing a single expected value for a future quantity, have historically dominated forecasting practice, their limitations in capturing uncertainty have become increasingly apparent. A more comprehensive understanding of the predictive distribution is essential for robust decision-making, particularly in domains characterized by high-impact, low-probability events. Quantile forecasts, which provide estimates of values below which a certain proportion of the distribution lies, offer a substantial enhancement in this regard. Quantile forecasts are indispensable for risk management and decision-making under uncertainty [41]. By providing a range of potential outcomes, they enable a more nuanced assessment of the likelihood of various scenarios. For instance, in financial markets, quantile forecasts can inform the calculation of value at risk (VaR), a crucial metric for risk management. In the energy sector, quantile forecasts of demand can assist in optimizing power generation and storage, mitigating the risks associated with demand fluctuations [42]. In the retail industry, quantile forecasts possess particular significance. Often, specific quantiles hold practical implications. For example, in a simplified supply chain context, the chosen quantile directly correlates with the optimal safety stock level in the newsvendor problem. By accurately estimating quantiles, retailers can mitigate stockouts and overstocks, thereby optimizing inventory levels and reducing costs. Moreover, quantile forecasts can inform pricing strategies, promotion planning, and demand forecasting for new products. Beyond these specific applications, quantile forecasts contribute to a more holistic approach to forecasting. They provide valuable insights into the shape of the predictive distribution, allowing for the identification of potential outliers, skewness, and other distributional characteristics. This information can be leveraged to improve the accuracy of point forecasts and to develop more sophisticated forecasting models [43]. In this study, all Transformer-based models were trained to minimize the Mean Absolute Error (MAE) for point forecasting and a Multi-Quantile Loss function (MQLoss) for probabilistic forecasting. The latter employed nine quantile levels: $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. The Transformer models were implemented using the `NeuralForecast` Python library [25].

5.4. Hyperparameter Tuning

Selecting a model that consistently excels in out-of-sample predictions is a critical step in the modeling process. To this end, a validation set is commonly employed to differentiate between competing models. Given the sensitivity of deep learning models to hyperparameter choices and initial conditions, a meticulous model selection strategy is indispensable. In this study, the final 28-day period of the training data, encompassing 29 February to 27 March 2016, was reserved as a validation set to objectively assess and rank alternative model configurations (please refer to Figure 2). To systematically explore the hyperparameter space and identify optimal configurations, the Optuna optimization framework [44] was employed.

Optuna is an open-source Python library for automating and accelerating hyperparameter optimization, particularly in machine learning. It helps find the best set of hyperparameters for models, improving performance. Optuna offers a define-by-run API for dynamic search space construction, efficient search algorithms like TPE, random search, and grid search, pruning for resource optimization, support for parallel and distributed optimization, and seamless integration with popular machine learning frameworks like TensorFlow, PyTorch, and scikit-learn. The process involves defining an objective function, creating a study, running optimization, and analyzing results. Optuna simplifies the often time-consuming task of hyperparameter tuning, allowing the researcher to focus on other machine learning aspects.

Table 2 outlines the hyperparameter search spaces explored in this study. As mentioned before, the Optuna hyperparameter optimization (HPO) framework was employed to randomly sample values from these spaces, generating various model configurations. The configuration yielding the highest validation score, as measured by mean absolute error (MAE) for point forecasts or multi-quantile loss (MQLoss) for probabilistic forecasts, was selected as the optimal model. In particular, the values for the learning rate were chosen randomly on a logarithmic scale between 10^{-4} and 10^{-2}, ensuring that each order of magnitude is equally represented. We first take the base-10 logarithm of the bounds to obtain -4 and -2, then sample a random value uniformly between these logarithmic bounds, and finally exponentiate the sampled value to transform it back to the original scale.

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

Hyperparameter	Range	Parameter Type
Input size	$\{28, 28 \times 2, 28 \times 3\}$	Discrete
Hidden size	$\{64, 128, 256\}$	Discrete
Learning rate	$[10^{-4}, 10^{-2}]$	Continuous (log)
Batch size	$\{32, 64, 128, 256\}$	Discrete
Windows batch size	$\{128, 256, 512, 1024\}$	Discrete
Random seed	$[1, 20]$	Discrete (int)

Table 3 summarizes the hyperparameter optimization settings for the Optuna framework. To mitigate overfitting, early stopping was implemented with a patience of four steps, and a validation check was performed every 100 steps. A total of 30 distinct model configurations were evaluated during the optimization process.

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

Parameter	Value
Maximum number of training steps	5000
Validation check steps	100
Early stopping patience steps	4
Number of trials	30
Validation function	MAE, MQLoss

Table 4 provides a detailed overview of the specific parameter configurations employed for each Transformer-based model in this study. These configurations were selected to ensure both optimal performance and comparability across different models. Key parameters include the number of multi-head self-attention layers, encoder layers, and decoder

layers, which vary across models, reflecting differences in their architectures. Other parameters, including convolutional hidden size, activation function, inference window batch size, and scaling, remain consistent across most models, ensuring uniformity in certain aspects of the network design. Unique parameters, such as the ProbSparse attention factor in the Informer and Autoformer models, the linear hidden size, patch length, and stride in PatchTST, and the moving average window in the Autoformer model, are specifically designed to enhance the efficiency and accuracy of these models under different scenarios.

Table 4. Specific parameter configurations for each Transformer-based model.

Parameter	Transformer	TFT	Informer	PatchTST	Autoformer
Multi-head self-attention layers	4	4	4	16	4
Encoder layers	2	—	2	3	2
Decoder layers	1	—	1	—	1
Convolutional hidden size	32	—	32	—	32
Activation function	GELU	—	GELU	GELU	GELU
Dropout	0.05	0.1	0.05	0.2	0.05
Attention layer dropout	—	0.0	—	0.0	—
Flatten head dropout	—	—	—	0.0	—
Linear layer dropout	—	—	—	0.2	—
Decoder input size multiplier	0.5	—	0.5	—	0.5
ProbSparse attention factor	—	—	3	—	3
Linear hidden size	—	—	—	256	—
Patch length	—	—	—	16	—
Stride	—	—	—	8	—
Moving average window	—	—	—	—	25
Inference windows batch size	1024	1024	1024	1024	1024
Scaling	Robust	Robust	Robust	Robust	Robust

Table 5 displays the optimal hyperparameter configurations determined by Optuna for both MAE (mean absolute error) and MQLoss (multi-quantile loss) optimization objectives. The upper and lower sections of the table correspond to MAE and MQLoss, respectively. Each model exhibited distinct optimal hyperparameter settings, highlighting the importance of tailoring hyperparameters to the specific architecture. Some trends were observed across multiple models. Most models benefited from larger input sizes, indicating the importance of capturing historical context. Hidden sizes varied across models, suggesting that the optimal size depends on the model's complexity and the nature of the data. Learning rates were generally low, demonstrating the need for careful optimization to avoid overfitting. Larger batch sizes and window batch sizes were common, suggesting that processing more data at once can improve training efficiency and stability. The random seed played a role in the optimal hyperparameters for some models, emphasizing the stochastic nature of the optimization process.

Table 5. Optimal hyperparameter configurations from Optuna: MAE (upper) and MQLoss (lower).

Hyperparameter	Transformer	TFT	Informer	PatchTST	Autoformer
MAE					
Input size	28×3	28×2	28×2	28	28
Hidden size	256	64	256	128	64
Learning rate	0.000527	0.003697	0.000160	0.000152	0.000349
Batch size	128	128	256	256	256
Windows batch size	512	1024	1024	128	1024
Random seed	20	4	4	5	8
MQLoss					
Input size	28×2	28×2	28×2	28×2	28
Hidden size	256	256	256	64	64
Learning rate	0.000114	0.000761	0.000290	0.001385	0.000862
Batch size	256	256	128	256	256
Windows batch size	512	1024	1024	512	1024
Random seed	1	18	16	8	17

5.5. Performance Measures

We evaluated the performance of Transformer-based models in both point and probabilistic forecasting contexts. To assess point forecast accuracy, we employed the mean absolute scaled error (MASE). MASE is computed by scaling the absolute forecast error by the historical seasonal error of the time series, as follows:

$$\mathrm{MASE}_{i,j} = \frac{\frac{1}{H}\sum_{t=L_j+1}^{L_j+H} |y_{i,t} - \hat{y}_{i,t}|}{\frac{1}{L_j - m}\sum_{t=m+1}^{L_j} |y_{i,t} - y_{i,t-1}|}, \tag{8}$$

$$\mathrm{MASE} = \frac{1}{N}\frac{1}{J}\sum_{i=1}^{N}\sum_{j=1}^{J} \mathrm{MASE}_{i,j}, \tag{9}$$

where $y_{i,t}$ is the value of time series i at time t, $\hat{y}_{i,t}$ is the corresponding forecast, H is the forecast horizon (28 days in this case), L_j is the forecast origin of cross-validation window j, J is the number of cross-validation windows (3 in this case), m is the seasonal period (7 days in this case), $\mathrm{MASE}_{i,j}$ is the MASE value for time series i on cross-validation window j, and N is the total number of time series (30,490 in our case study). This metric was chosen due to its robust properties compared to other point forecast evaluation measures [45].

The quality of probabilistic forecasts was assessed using the Weighted Quantile Loss (WQL) metric. Closely related to the Continuous Ranked Probability Score (CRPS), WQL is a standard measure for evaluating the accuracy of probabilistic forecasts [46,47]. It quantifies the agreement between the predicted probability distribution and the actual observations, across an evenly spaced grid of quantile levels [48]. Consistent with the MQLoss function, we computed the WQL using nine equally spaced quantile levels: $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$.

The Quantile Loss at level q of time series i at time t is defined as follows:

$$\rho_{q,i,t}(y_{i,t}, \hat{y}^q_{i,t}) = \begin{cases} 2(1-q)(\hat{y}^q_{i,t} - y_{i,t}), & \text{if } y_{i,t} < \hat{y}^q_{i,t} \\ 2q(y_{i,t} - \hat{y}^q_{i,t}), & \text{if } y_{i,t} \geq \hat{y}^q_{i,t} \end{cases}, \tag{10}$$

where $\hat{y}^{q}_{i,t}$ is the predicted quantile q of time series i at time t; that is, we expect the observation $y_{i,t}$ to be less than $\hat{y}^{q}_{i,t}$ with probability q. For example, the 10th predicted percentile would be $\hat{y}^{0.1}_{i,t}$. Thus, the quantile loss measures how well a model predicts a specific quantile of the actual distribution, asymmetrically penalizing overestimations and underestimations based on the quantile's value. The quantile loss at level q can be interpreted similarly to an absolute error. When $q = 0.5$, the quantile loss at level 0.5, $\rho_{0.5,i,t}(y_{i,t}, \hat{y}^{0.5}_{i,t})$ is equivalent to the absolute error. For other values of q, the "error" $(\hat{y}^{q}_{i,t} - y_{i,t})$ is weighted to reflect the likelihood of being positive or negative. If $q > 0.5$, $\rho_{q,i,t}(y_{i,t}, \hat{y}^{q}_{i,t})$ imposes a heavier penalty when the observation is greater than the estimated quantile than when it is less. The reverse is true for $q < 0.5$. In retail, the cost of being understocked is often higher than the cost of being overstocked. Therefore, forecasting at quantile $q = 0.75$ can be more informative than forecasting at the median quantile ($q = 0.50$). In these cases, $\rho_{0.75,i,t}(y_{i,t}, \hat{y}^{0.75}_{i,t})$ assigns a larger penalty weight to under-forecasting $(y_{i,t} \geq \hat{y}^{0.75}_{i,t})$ and a smaller penalty weight to over-forecasting $(y_{i,t} < \hat{y}^{0.75}_{i,t})$.

The weighted quantile loss for a cross-validation window j is calculated by dividing the total quantile loss by the sum of absolute time series values within the following forecast horizon:

$$\mathrm{WQL_j} = \frac{1}{q \sum_{i=1}^{N} \sum_{t=L_j+1}^{L_j+H} |y_{i,t}|} \sum_{i=1}^{N} \sum_{t=L_j+1}^{L_j+H} \sum_{q \in Q} \rho_{q,i,t}(y_{i,t}, \hat{y}^{q}_{i,t}) \,, \tag{11}$$

where Q represents the set of quantiles considered (in this case, $Q = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$), $\rho_{q,i,t}(y_{i,t}, \hat{y}^{q}_{i,t})$ is defined in Equation (10), and the other variables (H, L_j, and N) are defined as in the MASE equations (Equations (8) and (9)).

The overall weighted quantile loss is determined by taking the average of the WQL values across all cross-validation windows:

$$\mathrm{WQL} = \frac{1}{J} \sum_{j=1}^{J} \mathrm{WQL_j} \,, \tag{12}$$

where J is defined as in Equation (9).

By optimizing hyperparameters using MAE and MQLoss on a separate validation set, we mitigate the risk of overfitting, ensuring that the model generalizes well to unseen data. While MAE and MQLoss are effective for training and hyperparameter tuning, MASE and WQL are more robust metrics that account for scale invariance and distributional properties of the target variable, providing a more accurate assessment of real-world forecasting performance. This approach allows us to balance efficient model training with accurate prediction.

5.6. Results and Discussion

Tables 6 and 7 provide a comprehensive evaluation of Transformer-based models and traditional baselines across various forecast horizons. The models are assessed using mean absolute scaled error (MASE) and weighted quantile loss (WQL), with lower values indicating superior performance. The percentage values in these tables represent the improvement compared to the seasonal Naïve method.

Table 6 reveals that Transformer-based models significantly outperform baselines, particularly the seasonal Naïve method. Among the Transformer models, Transformer, Informer, and TFT exhibit the best performance, with consistent improvements in MASE ranging from approximately 26% to 29% across all forecast horizons. This indicates that these models effectively minimize absolute errors relative to a Naïve benchmark. While Autoformer and PatchTST also outperform baseline methods, their improvements are slightly lower, suggesting they may require further tuning to achieve optimal performance. In contrast, traditional baseline models like AutoARIMA and AutoETS, though better

than the seasonal Naïve method, lag significantly behind the Transformer-based models. The Naïve method, as expected, performs poorly, with positive percentages indicating a deterioration in accuracy compared to the seasonal Naïve approach.

Table 6. Performance of Transformer-based models and baselines evaluated with respect to MASE.

Forecasting Methods	**MASE**									
Transformer Models	Horizon = 1–7		Horizon = 8–14		Horizon = 15–21		Horizon = 22–28		Horizon = 1–28	
Transformer	0.8154	−29.02%	0.8874	−26.75%	0.8913	−26.49%	0.8806	−27.29%	0.8687	−27.37%
Informer	0.8164	−28.94%	0.8869	−26.80%	0.8901	−26.59%	0.8815	−27.21%	0.8687	−27.36%
TFT	0.8165	−28.92%	0.8867	−26.81%	0.8910	−26.51%	0.8813	−27.23%	0.8689	−27.35%
Autoformer	0.8344	−27.37%	0.9031	−25.46%	0.9080	−25.11%	0.9058	−25.21%	0.8878	−25.77%
PatchTST	0.8531	−25.74%	0.9215	−23.93%	0.9238	−23.81%	0.9138	−24.54%	0.9031	−24.49%
Baselines										
AutoARIMA	0.9741	−15.21%	1.0240	−15.48%	1.0286	−15.17%	1.0245	−15.41%	1.0128	−15.32%
AutoETS	0.9964	−13.27%	1.0619	−12.35%	1.0687	−11.86%	1.0653	−12.04%	1.0481	−12.37%
Seasonal Naïve	1.1488	—	1.2115	—	1.2125	—	1.2111	—	1.1960	—
Naïve	1.2511	+8.91%	1.3109	+8.20%	1.3190	+8.79%	1.3178	+8.81%	1.2997	+8.67%

Note: All percentage values in the table represent the percentage improvement compared to the seasonal Naïve method.

Table 7. Performance of Transformer-based models and baselines evaluated with respect to WQL.

Forecasting Methods	**WQL**									
Transformer Models	Horizon = 1–7		Horizon = 8–14		Horizon = 15–21		Horizon = 22–28		Horizon = 1–28	
Transformer	0.5298	−34.62%	0.5370	−30.72%	0.5477	−29.67%	0.5594	−30.80%	0.5436	−31.42%
Informer	0.5345	−34.05%	0.5410	−30.20%	0.5524	−29.06%	0.5628	−30.38%	0.5477	−30.89%
TFT	0.5294	−34.68%	0.5376	−30.63%	0.5496	−29.43%	0.5582	−30.94%	0.5438	−31.39%
Autoformer	0.5468	−32.52%	0.5508	−28.94%	0.5621	−27.82%	0.5751	−28.85%	0.5587	−29.50%
PatchTST	0.5365	−33.80%	0.5470	−29.42%	0.5602	−28.06%	0.5681	−29.72%	0.5531	−30.22%
Baselines										
AutoARIMA	0.5712	−29.52%	0.5670	−26.84%	0.5788	−25.69%	0.5934	−26.59%	0.5775	−27.14%
AutoETS	0.5649	−30.29%	0.5686	−26.63%	0.5797	−25.56%	0.5975	−26.07%	0.5777	−27.11%
Seasonal Naïve	0.8104	—	0.7750	—	0.7788	—	0.8083	—	0.7926	—
Naïve	0.8773	+8.26%	0.9630	+24.26%	1.0805	+38.74%	1.2385	+53.22%	1.0407	+31.30%

Note: All percentage values in the table represent the percentage improvement compared to the seasonal Naïve method.

Similarly, Table 7 shows that Transformer-based models substantially improve WQL over baseline methods. TFT and Transformer models again lead in performance, showing WQL reductions of up to 34% compared to the seasonal Naïve method, particularly for shorter forecast horizons (1–7 days). This underscores their effectiveness in minimizing forecast errors for probabilistic forecasting tasks. Informer also shows comparable performance, with slightly lower but still significant improvements. However, Autoformer and PatchTST exhibit smaller improvements in WQL, suggesting they may be less suited for minimizing quantile loss or require further refinement for this specific metric. The baseline models, AutoARIMA and AutoETS, show significant reductions in WQL, indicating better performance than the seasonal Naïve method but remaining inferior to Transformer-based models. Notably, the Naïve method performs poorly, with WQL increasing significantly, especially at longer horizons, indicating a substantial degradation in forecast quality.

An important aspect of the analysis is how the models' performance changes across different forecast horizons, as captured by MASE and WQL metrics. In Table 6, the variation of MASE across different horizons (1–7, 8–14, 15–21, 22–28 days) reveals interesting trends.

Transformer-based models exhibit their lowest MASE values in the shortest forecast horizon (1–7 days), highlighting their strong capability in accurately predicting short-term periods. As the forecast horizon extends, there is a slight increase in MASE across all

models, reflecting the inherent difficulty in making accurate predictions over longer periods. For instance, the Transformer model's MASE increases slightly from 0.8154 (1–7 days) to 0.8806 (22–28 days), maintaining substantial improvement over the baselines even in longer-term forecasting. Table 7 presents the variation in WQL across the same forecast horizons, offering insights into how models manage uncertainty in their predictions. Similar to MASE results, WQL values are lowest for Transformer models in the shortest horizon (1–7 days). For example, the TFT model achieves a WQL of 0.5294, demonstrating strong performance in handling short-term uncertainties. As the forecast horizon extends, WQL generally increases for all models, reflecting the increased difficulty in making accurate quantile predictions. However, the increase is relatively controlled for Transformer models, with the Transformer model's WQL rising modestly from 0.5298 (1–7 days) to 0.5594 (22–28 days). This suggests that these models retain their effectiveness in managing uncertainty, even as the forecasting task becomes more challenging. Baseline models, particularly AutoARIMA and AutoETS, also show an increase in WQL as the horizon extends, further highlighting their challenges in handling uncertainty in longer-term forecasts.

In Table 6, the percentage improvements in MASE compared to the seasonal Naïve method are detailed across various horizons. Transformer-based models show substantial improvements over the seasonal Naïve method, particularly in the short-term horizon. For instance, the Transformer model achieves a 29.02% reduction in MASE, indicating a significant enhancement in forecasting accuracy for short-term predictions. These improvements remain consistent as the forecast horizon extends to medium-term periods, with the Transformer models maintaining a percentage improvement close to 27–28%. This stability suggests that these models effectively reduce errors even as the task complexity increases with longer horizons. Although the improvement percentage slightly decreases as the forecast horizon extends to 22–28 days, the Transformer models still offer substantial MASE reductions, with the Transformer model achieving a 27.29% improvement, demonstrating its ability to outperform the seasonal Naïve baseline even in longer-term scenarios. Similarly, Table 7 provides insights into the percentage improvements in WQL compared to the seasonal Naïve method. The Transformer-based models show even greater improvements in WQL than in MASE. For example, the TFT model achieves a remarkable 34.68% reduction in WQL for the shortest horizon, highlighting its effectiveness in reducing uncertainty in short-term forecasts. As with MASE, improvements in WQL remain robust across medium-term horizons, with most Transformer models showing a 30–31% reduction. This indicates that the models not only maintain accuracy but also effectively manage uncertainty in predictions over these periods. Although the improvement in WQL slightly decreases as the forecast horizon extends, Transformer models still achieve a notable reduction. For instance, the Transformer model achieves a 30.80% improvement over the seasonal Naïve method, illustrating that even in more challenging, longer-term scenarios, these models significantly reduce uncertainty in their forecasts.

To better understand the relative performance, we calculated the relative MASE and relative WQL for each model compared to the seasonal Naïve method over the 1–28-day forecast horizon, as illustrated in Figure 3. This analysis provides a clearer picture of the percentage improvement achieved by each model. The relative performance analysis confirms the significant improvements achieved by Transformer-based models compared to the seasonal Naïve method and the baseline models. TFT, Informer, and Transformer exhibit the best overall performance across both metrics. While the Transformer models consistently outperform the baseline models in both MASE and WQL, it is worth noting that their relative performance is slightly different across the two metrics. The Transformer models exhibit larger relative improvements in MASE compared to the baseline models. This indicates that they are particularly effective in minimizing absolute errors. While the Transformer models still outperform the baseline models in WQL, their relative improvements are slightly smaller compared to MASE. This suggests that while they are effective in capturing uncertainty, their advantage over the baseline models may be less pronounced in terms of quantile loss.

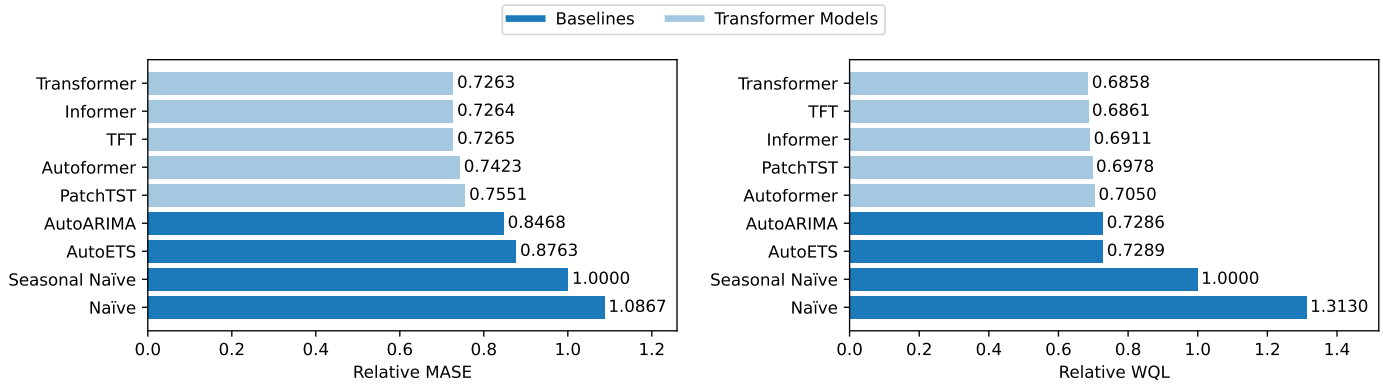

Figure 3. Relative MASE (**left**) and WQL (**right**) for Transformer-based models and baselines compared to the seasonal Naïve method over the 1–28-day forecast horizon.

Tables A1–A6 present a comprehensive comparison of the performance of Transformer-based models and baseline forecasting methods across different states, stores, categories, and departments, evaluated using the mean absolute scaled error (MASE) and weighted quantile loss (WQL) metrics. Table A1 presents the performance of Transformer-based models and baseline methods across three states, namely, California (CA), Texas (TX), and Wisconsin (WI). The table highlights the effectiveness of Transformer models, which generally outperform baseline methods across all states and metrics. However, the specific performance differences may vary slightly between states. This suggests that the effectiveness of Transformer models is not limited to a particular geographic region. Table A2 showcases the MASE for Transformer-based models and baselines across ten stores located in three states: California (CA1, CA2, CA3, and CA4), Texas (TX1, TX2, and TX3), and Wisconsin (WI1, WI2, and WI3). Table A3 illustrates the WQL for Transformer-based models and baselines across the same ten stores as in Table A2. The Transformer models exhibit consistent performance across all ten stores, indicating their ability to handle diverse store-specific characteristics. The baselines show more variability in performance, suggesting that their effectiveness may be more sensitive to store-level factors. Table A4 compares the performance of Transformer-based models and baselines across three categories—Foods, Hobbies, and Household—using both MASE and WQL metrics. The Transformer models again outperform the baselines across all three categories. However, the relative performance differences may vary slightly between categories. This suggests that the effectiveness of Transformer models is not limited to a particular product category. Table A5 evaluates the MASE for Transformer-based models and baselines across seven departments within the three main categories: Foods (Foods1, Foods2, and Foods3), Hobbies (Hobbies1 and Hobbies2), and Household (Household1 and Household2). Table A6 presents the WQL for the same departments as in Table A5. The Transformer models continue to exhibit superior performance across all seven departments. This highlights their ability to handle diverse product-specific characteristics.

To visualize the computational efficiency of Transformer-based models and baselines, Figure 4 presents a bar plot comparing their training and prediction times. When trained using the MAE, the Informer demonstrates the longest training time among the Transformer models, taking nearly 4 h, while PatchTST is the most efficient, completing training in just over 1 h. Other models like Autoformer, Transformer, and TFT fall in between these extremes, with training times ranging from approximately 2 h and 16 min to just under 3 h. In contrast, when trained using the MQLoss, all models experience a significant increase in training time, particularly TFT, which takes nearly 8 h. PatchTST remains the fastest but still requires more time compared to when using MAE, indicating the computational cost of training with MQLoss. Comparing these with the baseline methods, it is evident that traditional models like AutoARIMA are exceptionally time-consuming, with AutoARIMA taking over 24 h, far exceeding the time taken by even the slowest Transformer model. Simpler methods like Naïve and seasonal Naïve are much faster,

completing in mere seconds, which contrasts sharply with the more sophisticated but time-intensive Transformer models. Overall, the results underscore the trade-off between model complexity and computational efficiency, with Transformer-based models offering more sophisticated forecasting capabilities at the expense of significantly higher training times, especially when using more complex loss functions like MQLoss.

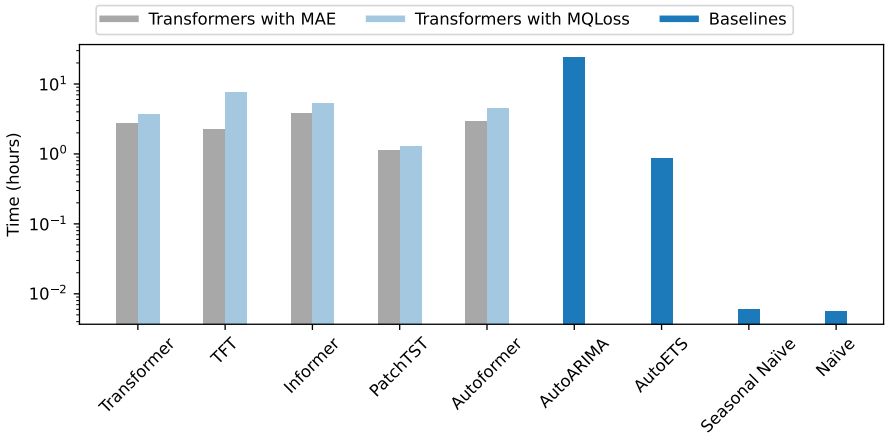

Figure 4. Training and prediction times for Transformer-based models using MAE and MQLoss, alongside baseline methods. The bar plot shows the computational efficiency of Transformer models and baselines, comparing the time required for training and prediction.

Figure 5 presents point forecasts on the left and probabilistic forecasts on the right for an illustrative time series from the Foods3 department of store TX3, across three cross-validation windows. These forecasts were generated by both Transformer-based models and traditional baselines. The forecasts are plotted alongside the ground truth values (gray), with the mean and median forecasts in blue and 80% prediction intervals shaded in light blue. Each plot includes MASE and WQL metrics to assess forecasting accuracy for each model.

Regarding the point forecasts, the Transformer-based models consistently outperform the baseline methods, demonstrating superior accuracy and effectively capturing the underlying patterns in the data. The forecast lines (blue) remain relatively stable, reflecting smoother predictions across the three cross-validation windows. Consequently, the MASE values for these models are lower, indicating better forecasting accuracy. The Autoformer model achieves the lowest MASE (1.040), closely followed by the Informer (1.054). AutoARIMA produces reasonable forecasts but exhibits some deviations from the ground truth, particularly after sharp spikes or dips, resulting in a higher MASE (1.534) compared to the Transformer-based models. AutoETS tends to underperform, as indicated by its higher MASE (2.554), reflecting its challenges in accurately predicting the actual values. The seasonal Naïve method struggles to capture the complexity of the time series, leading to pronounced oscillations in predictions that do not align well with the ground truth, as reflected by a MASE of 2.293. The Naïve method performs the worst, with a MASE of 3.035. Its simplistic approach results in predictions that often fail to capture any variations in the time series, leading to significant forecasting errors. Overall, the Transformer-based models outperform the traditional baselines in terms of MASE, with Autoformer and Informer showing particularly strong results. These models tend to provide more stable forecasts, which is beneficial for time series with less pronounced seasonal patterns or abrupt changes. Traditional baselines, particularly Naïve and seasonal Naïve, exhibit significant limitations in capturing the complexities of the time series, leading to poorer forecast accuracy. These findings highlight the effectiveness of Transformer-based models in forecasting tasks, especially compared to traditional time series models.

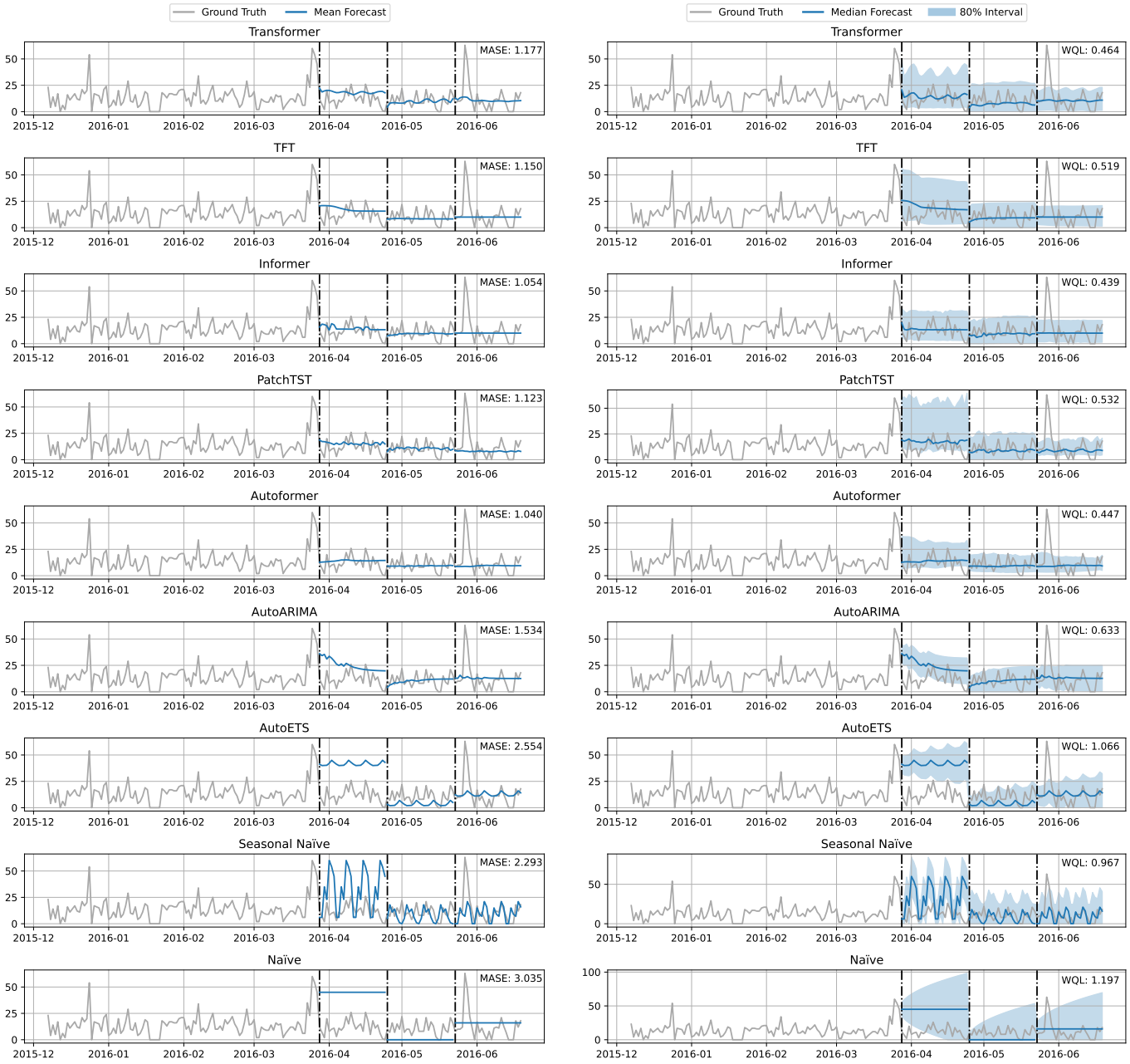

Figure 5. Point forecasts (**left**) and probabilistic forecasts (**right**) for an example series.

The plots on the right side of Figure 5 provide visual evidence of the uncertainty associated with the forecasts, as represented by the prediction intervals. The probabilistic forecasts of the Transformer-based models generally outperform those of the traditional baselines, with lower WQL values indicating better predictive distributions. Notably, the Informer and Autoformer models deliver the most accurate probabilistic forecasts within the Transformer-based category, achieving WQL values of 0.439 and 0.447, respectively, and generally exhibiting narrower prediction intervals. The Transformer, TFT, and PatchTST models follow with slightly higher WQL values (0.464, 0.519, and 0.532, respectively), suggesting marginally less accurate probabilistic forecasting. AutoARIMA performs relatively well among traditional models, with a WQL of 0.633, though it still lags behind the Transformer-based models. AutoETS and seasonal Naïve models exhibit higher WQL values of 1.066 and 0.967 respectively, indicating lower accuracy in their probabilistic predictions. The Naïve model has the highest WQL at 1.197, reflecting its simplistic approach and lack of sophistication in capturing the true distribution.

The plots in Figure 5 demonstrate the superior performance of Transformer-based models for time series forecasting in this particular case. However, the performance of these models may vary depending on the characteristics of the specific time series being forecasted. The choice of model may therefore depend on specific requirements, such as the need for accurate point forecasts or probabilistic predictions.

6. Conclusions

The findings of this study highlight the superior performance of Transformer-based models compared to traditional baseline methods in time series forecasting tasks. These models, particularly Transformer, Informer, and TFT, demonstrate significant improvements in both accuracy and probabilistic forecasting metrics. Specifically, these models achieve substantial reductions in mean absolute scaled error (MASE) and weighted quantile loss (WQL), outperforming the seasonal Naïve method and traditional models like AutoARIMA and AutoETS by wide margins. The consistent performance of Transformer, Informer, and TFT, with MASE improvements of 26% to 29% across all forecast horizons and WQL reductions of up to 34% for shorter horizons, underscores their effectiveness in minimizing forecasting errors. This suggests that these models are highly capable of handling diverse forecasting scenarios, offering robust performance across different evaluation metrics. While Autoformer and PatchTST also outperform baseline methods, their slightly lower performance and smaller improvements in WQL indicate that they may require further tuning or optimization, especially for probabilistic forecasting tasks. This highlights the importance of model selection and hyperparameter tuning in achieving the best possible results. This study also reveals a notable trade-off between model complexity and computational efficiency. Transformer-based models, while offering advanced forecasting capabilities, come with significantly higher training times, particularly when using complex loss functions like MQLoss. Informer, for instance, has the longest training time, while PatchTST is the most computationally efficient. Traditional models like AutoARIMA, however, are far more time-consuming, with training times exceeding 24 h, underscoring their inefficiency compared to the faster and more accurate Transformer-based models. In conclusion, Transformer-based models represent a significant advancement in time series forecasting, providing more accurate and reliable predictions than traditional methods. However, the increased computational demands of these models necessitate careful consideration of resource allocation and model optimization to balance forecasting performance with training efficiency.

The substantial improvements in forecasting accuracy achieved by Transformer-based models have practical implications for retail inventory management. By providing more accurate forecasts, these models can help retailers optimize inventory levels, reduce stockouts, and enhance overall operational efficiency. The adoption of these advanced models can lead to better decision-making processes and improved customer satisfaction through more reliable product availability. Our study underscores the potential of Transformer-based models as a significant advancement in time series forecasting. The findings suggest that retailers should consider integrating these models into their forecasting workflows to leverage their enhanced predictive capabilities. Moreover, the success of these models in the retail sector opens avenues for their application in other domains requiring accurate demand forecasting.

Nevertheless, applying Transformer models to retail data presents several challenges, primarily due to the large size of datasets typically encountered in this domain, often consisting of a vast number of time series. While the Transformer architecture has shown great promise in time series forecasting, it remains an area of active research. Numerous variations of the vanilla Transformer model have been developed, many of which are proving to be highly competitive and, as demonstrated in this study, even outperform traditional statistical approaches. However, one of the significant challenges associated with using Transformer models for retail forecasting is the algorithm's complexity. Transformers require considerable computational resources, both in terms of time and memory, especially

as the length of the input increases. This complexity can be a barrier to widespread adoption, particularly in environments where computational resources are limited. Despite these challenges, there is considerable excitement around the potential of Transformer models. Their advantages over other architectures, particularly in capturing complex temporal patterns [21], make them a compelling choice for future developments. We anticipate that ongoing research will continue to refine these models, making them more efficient and accessible for a broader range of applications in the near future.

Future research should explore the scalability and adaptability of Transformer-based models across different industries and datasets. Additionally, further investigation into optimizing model architectures, hyperparameter tuning, and training techniques could yield even greater forecasting performance. To further enhance the practical applicability of Transformer-based models in retail demand forecasting, future work could explore the inclusion of external covariates, such as promotions and holidays. Additionally, comparing these models to other advanced machine learning techniques, such as XGBoost and LightGBM, could provide valuable insights into their relative strengths and weaknesses. The ongoing development and refinement of these models promise to continue advancing the field of time series forecasting, offering robust solutions to complex predictive challenges.

Author Contributions: Conceptualization, J.M.O. and P.R.; methodology, J.M.O. and P.R.; software, J.M.O. and P.R.; validation, J.M.O. and P.R.; formal analysis, J.M.O. and P.R.; investigation, J.M.O. and P.R.; resources, J.M.O. and P.R.; data curation, J.M.O. and P.R.; writing—original draft preparation, J.M.O. and P.R.; writing—review and editing, J.M.O. and P.R.; visualization, J.M.O. and P.R. All authors have read and agreed to the published version of the manuscript.

Funding: This work is financed by Portuguese national funds through FCT—Fundação para a Ciência e Tecnologia, under the project UIDP/05422/2020.

Data Availability Statement: A publicly available dataset was used in this study. The data can be found here: https://www.kaggle.com/competitions/m5-forecasting-accuracy/data (accessed on 1 April 2024).

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

Table A1. Performances of Transformer-based models and baselines across three states: California (CA), Texas (TX), and Wisconsin (WI); evaluated with respect to MASE and WQL.

Forecasting Methods		**MASE**			**WQL**	
Transformer Models	CA	TX	WI	CA	TX	WI
Transformer	0.8900	0.8396	0.8694	0.5498	0.5430	0.5355
Informer	0.8907	0.8399	0.8683	0.5543	0.5468	0.5396
TFT	0.8912	0.8397	0.8683	0.5503	0.5422	0.5363
Autoformer	0.9100	0.8585	0.8877	0.5643	0.5590	0.5508
PatchTST	0.9232	0.8758	0.9036	0.5594	0.5542	0.5435
Baselines						
AutoARIMA	1.0166	0.9982	1.0222	0.5821	0.5827	0.5667
AutoETS	1.0505	1.0383	1.0547	0.5797	0.5887	0.5655
Seasonal Naïve	1.2114	1.1764	1.1950	0.7999	0.8220	0.7571
Naïve	1.3611	1.2564	1.2611	1.0741	1.0856	0.9560

Table A2. Performances of Transformer-based models and baselines across ten stores located in three states: California (CA1, CA2, CA3, and CA4), Texas (TX1, TX2, and TX3), and Wisconsin (WI1, WI2, and WI3); evaluated with respect to MASE.

Forecasting Methods	**MASE**									
Transformer Models	CA1	CA2	CA3	CA4	TX1	TX2	TX3	WI1	WI2	WI3
Transformer	0.8186	0.9901	0.8665	0.8846	0.9058	0.7514	0.8615	0.8893	0.8888	0.8301
Informer	0.8194	0.9903	0.8685	0.8845	0.9064	0.7514	0.8618	0.8895	0.8865	0.8289
TFT	0.8192	0.9896	0.8709	0.8852	0.9064	0.7515	0.8612	0.8882	0.8864	0.8304
Autoformer	0.8374	1.0102	0.8876	0.9046	0.9247	0.7694	0.8812	0.9090	0.9070	0.8470
PatchTST	0.8487	1.0248	0.8955	0.9237	0.9441	0.7869	0.8963	0.9257	0.9212	0.8638
Baselines										
AutoARIMA	0.9331	1.1078	0.9760	1.0497	1.0785	0.8957	1.0205	1.0264	1.0372	1.0030
AutoETS	0.9496	1.1139	1.0048	1.1336	1.1275	0.9276	1.0597	1.0531	1.0811	1.0299
Seasonal Naïve	1.1042	1.3177	1.1663	1.2576	1.2618	1.0634	1.2040	1.2148	1.2202	1.1498
Naïve	1.2048	1.5542	1.3096	1.3759	1.3922	1.1410	1.2362	1.3334	1.2359	1.2140

Table A3. Performances of Transformer-based models and baselines across ten stores located in three states: California (CA1, CA2, CA3, and CA4), Texas (TX1, TX2, and TX3), and Wisconsin (WI1, WI2, and WI3); evaluated with respect to WQL.

Forecasting Methods	**WQL**									
Transformer Models	CA1	CA2	CA3	CA4	TX1	TX2	TX3	WI1	WI2	WI3
Transformer	0.5443	0.5606	0.5130	0.6235	0.5508	0.5369	0.5423	0.5583	0.5173	0.5376
Informer	0.5498	0.5659	0.5171	0.6262	0.5552	0.5408	0.5455	0.5626	0.5206	0.5424
TFT	0.5467	0.5631	0.5116	0.6218	0.5501	0.5368	0.5409	0.5587	0.5174	0.5396
Autoformer	0.5594	0.5754	0.5257	0.6411	0.5674	0.5526	0.5583	0.5736	0.5321	0.5534
PatchTST	0.5534	0.5719	0.5193	0.6389	0.5633	0.5473	0.5532	0.5675	0.5244	0.5455
Baselines										
AutoARIMA	0.5793	0.5886	0.5410	0.6688	0.5947	0.5808	0.5744	0.5942	0.5421	0.5725
AutoETS	0.5749	0.5787	0.5426	0.6734	0.6035	0.5810	0.5837	0.5874	0.5443	0.5724
Seasonal Naïve	0.7927	0.7873	0.7564	0.9322	0.8483	0.8224	0.7990	0.8001	0.7120	0.7752
Naïve	1.0621	1.0867	1.0154	1.2056	1.1568	1.0966	1.0135	1.0442	0.8525	1.0075

Table A4. Performances of Transformer-based models and baselines across three categories: Foods, Hobbies, and Household; evaluated with respect to MASE and WQL.

Forecasting Methods	**MASE**			**WQL**		
Transformer Models	Foods	Hobbies	Household	Foods	Hobbies	Household
Transformer	0.8932	0.7915	0.8767	0.5099	0.7007	0.5772
Informer	0.8921	0.7933	0.8774	0.5145	0.7028	0.5811
TFT	0.8915	0.7942	0.8782	0.5109	0.6991	0.5760
Autoformer	0.9110	0.8116	0.8972	0.5231	0.7215	0.5960
PatchTST	0.9219	0.8344	0.9143	0.5173	0.7186	0.5896
Baselines						
AutoARIMA	0.9970	1.0041	1.0391	0.5366	0.7673	0.6191
AutoETS	1.0075	1.0841	1.0843	0.5372	0.7718	0.6164
Seasonal Naïve	1.1799	1.1889	1.2218	0.7317	1.0869	0.8497
Naïve	1.2817	1.2845	1.3326	0.9576	1.4460	1.1171

Table A5. Performances of Transformer-based models and baselines across seven departments—three from the Foods category (Foods1, Foods2, and Foods3), two from the Hobbies category (Hobbies1 and Hobbies2), and two from the Household category (Household1 and Household2)—evaluated with respect to MASE.

Forecasting Methods	**MASE**						
Transformer Models	Foods1	Foods2	Foods3	Hobbies1	Hobbies2	Household1	Household2
Transformer	0.8751	0.9367	0.8769	0.8054	0.7529	0.9637	0.7867
Informer	0.8738	0.9341	0.8766	0.8070	0.7549	0.9634	0.7885
TFT	0.8748	0.9339	0.8754	0.8080	0.7558	0.9640	0.7895
Autoformer	0.8906	0.9537	0.8956	0.8247	0.7751	0.9858	0.8056
PatchTST	0.9043	0.9683	0.9040	0.8470	0.7992	0.9954	0.8306
Baselines							
AutoARIMA	1.0009	1.0539	0.9685	0.9911	1.0405	1.0593	1.0182
AutoETS	1.0245	1.0674	0.9740	1.0648	1.1380	1.0783	1.0906
Seasonal Naïve	1.2174	1.2380	1.1420	1.1737	1.2314	1.2462	1.1966
Naïve	1.2792	1.3422	1.2531	1.2901	1.2690	1.3667	1.2975

Table A6. Performances of Transformer-based models and baselines across seven departments—three from the Foods category (Foods1, Foods2, and Foods3), two from the Hobbies category (Hobbies1 and Hobbies2), and two from the Household category (Household1 and Household2)—evaluated with respect to WQL.

Forecasting Methods	**WQL**						
Transformer Models	Foods1	Foods2	Foods3	Hobbies1	Hobbies2	Household1	Household2
Transformer	0.5858	0.5530	0.4847	0.6790	0.8909	0.5302	0.7504
Informer	0.5875	0.5570	0.4900	0.6809	0.8943	0.5339	0.7551
TFT	0.5840	0.5545	0.4860	0.6776	0.8885	0.5293	0.7481
Autoformer	0.5963	0.5669	0.4981	0.6973	0.9338	0.5464	0.7889
PatchTST	0.5955	0.5652	0.4903	0.6956	0.9210	0.5395	0.7743
Baselines							
AutoARIMA	0.6138	0.5852	0.5095	0.7343	1.0575	0.5578	0.8449
AutoETS	0.6259	0.5877	0.5076	0.7357	1.0887	0.5515	0.8557
Seasonal Naïve	0.8614	0.7837	0.6948	1.0330	1.5596	0.7624	1.1714
Naïve	1.0774	1.0113	0.9219	1.3824	2.0038	1.0028	1.5389

References

1. Wellens, A.P.; Boute, R.N.; Udenio, M. Simplifying tree-based methods for retail sales forecasting with explanatory variables. *Eur. J. Oper. Res.* **2024**, *314*, 523–539. [CrossRef]
2. Ramos, P.; Oliveira, J.M.; Kourentzes, N.; Fildes, R. Forecasting Seasonal Sales with Many Drivers: Shrinkage or Dimensionality Reduction? *Appl. Syst. Innov.* **2023**, *6*, 3. [CrossRef]
3. Petropoulos, F.; Apiletti, D.; Assimakopoulos, V.; Babai, M.Z.; Barrow, D.K.; Ben Taieb, S.; Bergmeir, C.; Bessa, R.J.; Bijak, J.; Boylan, J.E.; et al. Forecasting: Theory and practice. *Int. J. Forecast.* **2022**, *38*, 705–871. [CrossRef]
4. Oliveira, J.M.; Ramos, P. Investigating the Accuracy of Autoregressive Recurrent Networks Using Hierarchical Aggregation Structure-Based Data Partitioning. *Big Data Cogn. Comput.* **2023**, *7*, 100. [CrossRef]
5. Wen, Q.; Zhou, T.; Zhang, C.; Chen, W.; Ma, Z.; Yan, J.; Sun, L. Transformers in Time Series: A Survey. In Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23, Macao, China, 19–25 August 2023 ; Elkind, E., Ed.; Survey Track; International Joint Conferences on Artificial Intelligence Organization: Darmstadt, Germany, 2023; pp. 6778–6786. [CrossRef]
6. Ahmed, S.; Nielsen, I.E.; Tripathi, A.; Siddiqui, S.; Ramachandran, R.P.; Rasool, G. Transformers in Time-Series Analysis: A Tutorial. *Circuits Syst. Signal Process.* **2023**, *42*, 1531–5878. [CrossRef]
7. Lindemann, B.; Müller, T.; Vietz, H.; Jazdi, N.; Weyrich, M. A survey on long short-term memory networks for time series prediction. *Procedia CIRP* **2021**, *99*, 650–655. [CrossRef]
8. Zeng, A.; Chen, M.; Zhang, L.; Xu, Q. Are Transformers Effective for Time Series Forecasting? *Proc. AAAI Conf. Artif. Intell.* **2023**, *37*, 11121–11128. [CrossRef]

9. Li, S.; Jin, X.; Xuan, Y.; Zhou, X.; Chen, W.; Wang, Y.X.; Yan, X. Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting. *arXiv* **2019**, arXiv:1907.00235.
10. Mahmoud, A.; Mohammed, A. A Survey on Deep Learning for Time-Series Forecasting. In *Machine Learning and Big Data Analytics Paradigms: Analysis, Applications and Challenges*; Hassanien, A.E., Darwish, A., Eds.; Springer International Publishing: Cham, Switzerland, 2021; pp. 365–392. [CrossRef]
11. Benidis, K.; Rangapuram, S.S.; Flunkert, V.; Wang, Y.; Maddix, D.; Turkmen, C.; Gasthaus, J.; Bohlke-Schneider, M.; Salinas, D.; Stella, L.; et al. Deep Learning for Time Series Forecasting: Tutorial and Literature Survey. *ACM Comput. Surv.* **2022**, *55*, 1–36. [CrossRef]
12. Oliveira, J.M.; Ramos, P. Cross-Learning-Based Sales Forecasting Using Deep Learning via Partial Pooling from Multi-level Data. In *Proceedings of the Engineering Applications of Neural Networks*; Iliadis, L., Maglogiannis, I., Alonso, S., Jayne, C., Pimenidis, E., Eds.; Springer Nature: Cham, Switzerland, 2023; pp. 279–290.
13. Casolaro, A.; Capone, V.; Iannuzzo, G.; Camastra, F. Deep Learning for Time Series Forecasting: Advances and Open Problems. *Information* **2023**, *14*, 598. [CrossRef]
14. Miller, J.A.; Aldosari, M.; Saeed, F.; Barna, N.H.; Rana, S.; Arpinar, I.; Liu, N. A Survey of Deep Learning and Foundation Models for Time Series Forecasting. *arXiv* **2024**, arXiv:2401.13912.
15. Ramos, P.; Oliveira, J.M. Robust Sales forecasting Using Deep Learning with Static and Dynamic Covariates. *Appl. Syst. Innov.* **2023**, *6*, 85. [CrossRef]
16. Lim, B.; Zohren, S. Time-series forecasting with deep learning: A survey. *Philos. Trans. R. Soc. A Math. Phys. Eng. Sci. R. Soc.* **2021**, *379*, 20200209. [CrossRef]
17. Lin, T.; Wang, Y.; Liu, X.; Qiu, X. A survey of transformers. *AI Open* **2022**, *3*, 111–132. [CrossRef]
18. Padhi, I.; Schiff, Y.; Melnyk, I.; Rigotti, M.; Mroueh, Y.; Dognin, P.; Ross, J.; Nair, R.; Altman, E. Tabular Transformers for Modeling Multivariate Time Series. In Proceedings of the ICASSP 2021—2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Toronto, ON, Canada, 6–11 June 2021; pp. 3565–3569. [CrossRef]
19. Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; Zhang, W. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. *Proc. AAAI Conf. Artif. Intell.* **2021**, *35*, 11106–11115. [CrossRef]
20. Wu, H.; Xu, J.; Wang, J.; Long, M. Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 22419–22430.
21. Lim, B.; Arık, S.Ö.; Loeff, N.; Pfister, T. Temporal Fusion Transformers for interpretable multi-horizon time series forecasting. *Int. J. Forecast.* **2021**, *37*, 1748–1764. [CrossRef]
22. Nie, Y.; Nguyen, N.H.; Sinthong, P.; Kalagnanam, J. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In Proceedings of the Eleventh International Conference on Learning Representations, Kigali, Rwanda, 1–5 May 2023.
23. Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; et al. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *arXiv* **2020**, arXiv:1910.03771.
24. Erickson, N.; Mueller, J.; Shirkov, A.; Zhang, H.; Larroy, P.; Li, M.; Smola, A. AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data. *arXiv* **2020**, arXiv:2003.06505.
25. Olivares, K.G.; Challú, C.; Garza, F.; Canseco, M.M.; Dubrawski, A. *NeuralForecast: User Friendly State-of-the-Art Neural Forecasting Models*; PyCon: Salt Lake City, UT, USA, 2022.
26. Liu, S.; Yu, H.; Liao, C.; Li, J.; Lin, W.; Liu, A.X.; Dustdar, S. Pyraformer: Low-Complexity Pyramidal Attention for Long-Range Time Series Modeling and Forecasting. In Proceedings of the International Conference on Learning Representations, Virtual, 25 April 2022.
27. Zhou, T.; Ma, Z.; Wen, Q.; Wang, X.; Sun, L.; Jin, R. FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting. In Proceedings of the 39th International Conference on Machine Learning, Baltimore, MD, USA, 17–23 July 2022; Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., Sabato, S., Eds.; Proceedings of Machine Learning Research; PMLR: Cambridge, MA, USA, 2022; Volume 162, pp. 27268–27286.
28. Liu, Y.; Wu, H.; Wang, J. Non-stationary transformers: Exploring the stationarity in time series forecasting. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 9881–9893.
29. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.u.; Polosukhin, I. Attention is All you Need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Volume 30, pp. 5998–6008.
30. Xiong, R.; Yang, Y.; He, D.; Zheng, K.; Zheng, S.; Xing, C.; Zhang, H.; Lan, Y.; Wang, L.; Liu, T.Y. On layer normalization in the transformer architecture. In Proceedings of the 37th International Conference on Machine Learning, Online, 13–18 July 2020.
31. Oliveira, J.M.; Ramos, P. Assessing the Performance of Hierarchical Forecasting Methods on the Retail Sector. *Entropy* **2019**, *21*, 436. [CrossRef] [PubMed]
32. Ramos, P.; Santos, N.; Rebelo, R. Performance of state space and ARIMA models for consumer retail sales forecasting. *Robot. Comput.-Integr. Manuf.* **2015**, *34*, 151–163. [CrossRef]
33. Ramos, P.; Oliveira, J.M. A procedure for identification of appropriate state space and ARIMA models based on time-series cross-validation. *Algorithms* **2016**, *9*, 76. [CrossRef]
34. Garza, F.; Canseco, M.M.; Challú, C.; Olivares, K.G. *StatsForecast: Lightning Fast Forecasting with Statistical and Econometric Models*; PyCon: Salt Lake City, UT, USA, 2022.

35. Hyndman, R.J.; Khandakar, Y. Automatic time series forecasting: The forecast package for R. *J. Stat. Softw.* **2008**, *27*, 1–22. [CrossRef]
36. Hyndman, R.J.; Koehler, A.B.; Snyder, R.D.; Grose, S. A state space framework for automatic forecasting using exponential smoothing methods. *Int. J. Forecast.* **2002**, *18*, 439–454. [CrossRef]
37. Hyndman, R.J.; Koehler, A.B.; Ord, J.K.; Snyder, R.D. *Forecasting with Exponential Smoothing: The State Space Approach*; Springer Series in Statistics; Springer: Berlin, Germany, 2008. [CrossRef]
38. Hyndman, R.J.; Athanasopoulos, G. *Forecasting: Principles and Practice*, 3rd ed.; OTexts: Melbourne, Australia, 2021.
39. Ord, J.K.; Fildes, R.; Kourentzes, N. *Principles of Business Forecasting*, 2nd ed.; Wessex Press Publishing Co.: London, UK, 2017.
40. Makridakis, S.; Spiliotis, E.; Assimakopoulos, V. The M5 competition: Background, organization, and implementation. *Int. J. Forecast.* **2022**, *38*, 1325–1336. [CrossRef]
41. Koenker, R. *Quantile Regression*; Econometric Society Monographs; Cambridge University Press: Cambridge, UK, 2005.
42. Eisenach, C.; Patel, Y.; Madeka, D. MQTransformer: Multi-Horizon Forecasts with Context Dependent and Feedback-Aware Attention. *arXiv* **2022**, arXiv:2009.14799.
43. Wen, R.; Torkkola, K.; Narayanaswamy, B.; Madeka, D. A Multi-Horizon Quantile Recurrent Forecaster. *arXiv* **2018**, arXiv:1711.11053.
44. Akiba, T.; Sano, S.; Yanase, T.; Ohta, T.; Koyama, M. Optuna: A Next-generation Hyperparameter Optimization Framework. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Anchorage, AK, USA, 4–8 August 2019.
45. Hyndman, R.J.; Koehler, A.B. Another look at measures of forecast accuracy. *Int. J. Forecast.* **2006**, *22*, 679–688. [CrossRef]
46. Gneiting, T.; Raftery, A.E. Strictly Proper Scoring Rules, Prediction, and Estimation. *J. Am. Stat. Assoc.* **2007**, *102*, 359–378. [CrossRef]
47. Gasthaus, J.; Benidis, K.; Wang, Y.; Rangapuram, S.S.; Salinas, D.; Flunkert, V.; Januschowski, T. Probabilistic Forecasting with Spline Quantile Function RNNs. In Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics, Naha, Okinawa, Japan, 16–18 April 2019; Chaudhuri, K., Sugiyama, M., Eds.; Proceedings of Machine Learning Research; PMLR: Cambridge, MA, USA, 2019; Volume 89, pp. 1901–1910.
48. Shchur, O.; Turkmen, C.; Erickson, N.; Shen, H.; Shirkov, A.; Hu, T.; Wang, Y. AutoGluon-TimeSeries: AutoML for Probabilistic Time Series Forecasting. In Proceedings of the International Conference on Automated Machine Learning, Potsdam/Berlin, Germany, 12–15 September 2023; PMLR: Cambridge, MA, USA, 2023; pp. 1–21.