Master's thesis
Master's Programme in Data Science

Modern time series methods for demand forecasting in retail

Jarkko Rahikka

June 23, 2025

Supervisor(s): Dr. Petrus Mikkola

Examiner(s): Professor Arto Klami
Dr. Petrus Mikkola

University of Helsinki
Faculty of Science

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Degree programme
Faculty of Science	Master's Programme in Data Science

Tekijä — Författare — Author

Jarkko Rahikka

Työn nimi — Arbetets titel — Title

Modern time series methods for demand forecasting in retail

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidantal — Number of pages
Master's thesis	June 23, 2025	55

Tiivistelmä — Referat — Abstract

The retail industry is a dynamic industry, with a large number of retailers relying on human labor and typically low profit margins. Accurate forecasting of demand is crucial for most decisions at the strategic, tactical, and operational levels, such as market selection, assortment decisions, and replenishment. For years, traditional statistical methods have been the popular choice for time-series forecasting, but lately machine learning has emerged as a transformative technology, enabling computers to learn patterns and make decisions without explicit programming. Machine learning has also been adapted for time series forecasting cases, and lately Transformer-based models have emerged as one of the most promising models for accurate time series forecasting. In this study, I will evaluate how well modern Transformer-based models can perform in a retail environment, what are their strengths and weaknesses, and also how other models perform compared to them. The study is performed as a literature review of selected scientific articles on the topic. The study answers the following question: Are Transformer-based models capable of solving demand forecasting problems in retail?

First, I introduce the retail industry and some examples of problems where demand forecasting is needed, with a short overview of the articles used in the study. The terms and formulas are explained in Chapter 2, with an overview of models used for time series forecasting in retail. In addition, different variants of Transformer-based models are introduced. In Chapter 3, I discuss the strengths and weaknesses of the different methods in retail sales forecasting. The focus is on Transformer-based methods and how they compare with other models. The chapter starts with a deeper look into the requirements in a retail environment and how the best models perform in general. Then, the latest Transformer-based models are evaluated in the retail environment based on accuracy, complexity, and interpretability. Chapter 4 brings it all together. First, I summarize the findings of the study Chapter by Chapter, especially concentrating on the strengths and weaknesses of Transformer-based models in retail time series forecasting. When measured by accuracy, Transformer-based models seem to be superior compared to other models, especially the traditional statistical ones. Their weaknesses lie in their complexity, as they still need a lot of time and space to train. In addition, interpretability is not one of their strengths. The final chapter ends with some suggestions for further studies.

ACM Computing Classification System (CCS):
General and reference → Document types → Surveys and overviews
Applied computing → Document management and text processing → Document management → Text editing

Avainsanat — Nyckelord — Keywords

demand forecasting, sales forecasting, time series, Transformers

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Contents

1. Introduction

Retail refers to the sale of goods and services to consumers through on-line and offline sales channels. A retailer purchases products from suppliers and sells them through selected channels in order to generate a profit. Large retailers can operate across multiple countries, managing thousands of stores and offering a vast assortment of products both online and offline. The retail sector is highly competitive, which requires efficient management of core processes to remain profitable, as margins are often low. Given the industry's reliance on human labor and typically low profit margins, leveraging machine learning (ML) applications presents an opportunity to improve operational efficiency and profitability. [40]

At the strategic level, retail management focuses on shaping the future of the business. This involves determining the target markets, identifying segments of customers to serve, and defining the product and service offerings. Key decisions include the selection of distribution channels, store formats, and optimal locations. A well-defined pricing policy provides guidance for tactical pricing at the operational level, and all of these decisions should be based on accurate forecasts of future demand.

One of the key tactical level challenges in retail is optimizing product assortment. Due to limited shelf space and potential cannibalization between similar products, offering the full range in all stores is not feasible. Various algorithms are available for assortment optimization, and accurate sales forecasts are the basis for all of them [29]. Closely related to assortment optimization is inventory management, where the central question is how much stock should be maintained at different points in the supply chain at any given time. Both assortment optimization and inventory management are based on accurate demand forecasts to ensure efficiency and profitability.

At the operational level, demand forecasts are needed for out-of-stock management, supplier analysis, replenishment, and procurement [5]. The aim of a retailer is to manage the inventory of various items so that there is no overstocking and no out-of-stock situations. The optimization of replenishment with the use of ML is the key here [40], and decisions are based on accurate forecasts of future demand. At the store-level, forecasts for replenishment are often calculated at the daily level.

Having accurate, reliable, and fast sales predictions is crucial for the success

of a retailer. In addition, to be able to compete in a highly competitive industry, the methods used must be competitive with the methods used by the rivals. This competitive environment has led to an enormous effort in developing better and better methods to predict the demand for each individual product sold.

Traditional statistical forecasting methods have been used for decades to make better decisions in retail, and in the past years different kind of machine learning forecasting methods have emerged to challenge them, as the most recent Transformer-based models. The amount of available data and computing power has increased dramatically, and traditional sales forecasting methods simply cannot take full advantage of the enormous amount of data. Classical forecasting methods are able to use only the sales history when making forecasts, and this is clearly one of the limitations.

Traditional time series forecasting methods often get into trouble when dealing with complex nonlinear patterns, multiple seasonalities, different kinds of trend, various external factors, and cross-relationships of sales of different products, which are all factors typical to retail data. Modern methods introduced in recent years are better capable of handling such complex relationships and have shown great potential for use in the retail environment. Deep learning methods, especially Transformer-based models, are one of the most promising types of modern methods for time series forecasting.

1.1 The research question and the methodology

Since sales forecasting serves as a base for several critical business decisions, in this study, I will present a literature review of the different forecasting methods for retail sales. The aim is to find out which methods are currently performing best and what the main advantages and limitations of the different methods are. The focus is on the newest methods like Transformers, but traditional models will also be discussed as needed. The main focus will be on the strategic and tactical level decision-making, where long-term multistep forecasts are needed, but I will discuss strengths and weaknesses of each model for different use-cases, since probably there is not one model that is best in each possible forecasting case.

The main research question is "are Transformer-based models capable of solving problems of demand forecasting in retail", and what their advantages and disadvantages are compared to other methods. In addition, an overview of the development of time-series methods, especially Transformers, is given.

The study was carried out as a literature research with a focus on the latest models and methods, including general articles on machine learning in retail applications, several articles about time series forecasting, and finally the latest articles about Transformer-based models in retail demand forecasting. The articles were selected from

Table 1.1: The table shows the number of articles used for each year.

Year	-2020	2021	2022	2023	2024	2025
Number of articles	6	2	3	6	28	5

peer-reviewed scientific papers available from different online libraries. More than a hundred articles were previewed before choosing the ones to actually use in this study, which cover retail forecasting and the use of Transformer-based models pretty well.

Articles like [5, 40, 29] were used to learn about different machine learning applications in retail, and an overview of the sales forecasting problem with some solutions was given in articles like [12, 27]. Time series forecasting in general, with forecasting models, was introduced in articles such as [1, 3, 6, 21, 44, 46, 24, 32, 48, 39]. Most of the articles introduced a new model with some comparisons to older models. Articles like [8, 9, 16, 17, 18, 20, 21, 26, 30, 31, 37, 38, 42, 45, 50] belong to this group. Articles specialized in comparing models for probabilistic forecasting included [7, 33] and articles focused on a narrower problem in retailing included [4] on store segmentation, [9] on prediction of e-commerce sales, [13] on store clustering, [15] on impact of in-store displays, [21] about dynamic pricing and inventory management. Sales forecasting methods for some special cases were introduced in [35, 34] (fashion products), [36] (promotional effects), [41] (explanatory variables and tree-based methods) and [43, 14] (markdown optimization). Forecasting drugstore sales with a Transformer-based fusion model was introduced in [47]. Sales forecasting for new products was handled in [34, 31]. Articles [11] and [19] were about foundation models in forecasting. One article focused solely on the interpretability of deep learning models [2].

There are many more models available than those mentioned in this study, but I tried to select the most important ones. Since the topic area is evolving with enormous speed, one criterion was to choose articles that have been published recently. The distribution is given in Table (see Table 1.1). As can be seen, most articles are from the year 2024, and some even from 2025.

1.2 Structure of the study

Chapter 2 starts with definitions for time-series forecasting problems, added with formulas for the loss functions used to assess the accuracy of different models. After that, I give a short introduction into different time series forecasting methods, from traditional statistical models to the modern Transformer-based models. The section on Transformer-based models tries to give a chronological look at some of the best variants and their main functionalities.

In Chapter 3, I go deeper into the forecasting problems in retail. The chapter starts with introducing different types of problem, what kind of data is available for retailers, and what features would make a good forecasting model. Then I discuss the performance of the selected models in more depth, first in general and then in the context of retail business. Traditional forecasting methods are compared with several modern Transformer-based models, and the assessment is performed based on accuracy, complexity, and usability. The focus is on Transformer-based models for time series forecasting in the retail environment and how they perform in different forecasting tasks compared to other methods. Transformer-based models will also be compared with each other to find out which kind of functionalities are behind the best results.

Chapter 4 will bring it all together with some practical suggestions and possible future study topics. Firs, I briefly summarize the results of the previous chapters, draw some conclusions of modern methods for time series forecasting in a retail environment, and finally give some suggestions for further study topics.

2. Background and definitions

Before going deeper into the descriptions of the different methods for time-series forecasting, it is useful to introduce some basic terms and formulas that will be used later in the text. I start with some basic definitions of time series, then I introduce some loss functions for evaluating the accuracy of the models, and in the end of this chapter, I go through the development steps in time series forecasting from the early statistical methods to the latest Transformer-based deep learning models.

2.1 Time series forecasting definitions

A time series is a sequence of data points recorded over time at regular intervals. In a univariate version, it captures the evolution of one variable over time, making it essential for analyzing trends, seasonal patterns, and future predictions. Sales data is one example of time series data, where historical sales are recorded for several time points in the past. For a univariate time series X, with a look-back window size T, and a forecast horizon L, the forecasting problem can be written [24]:

$$X = (x_1, x_2, ..., x_T); Y = (y_1, y_2, ..., y_{T+L}). \tag{2.1}$$

The forecast can be expressed as T x 1 $\rightarrow$ L x 1, where T is the number of past values x_T and L is the number of predicted values y, and it is also worth mentioning that this is a special case of the multivariate problem. In the multivariate case, we have multiple variables both in the past and in the future values, and the problem changes to [24]

$$X = \{X_1^t, X_2^t, ..., X_M^t\}_{t=1}^{T}; Y = \{Y_1^t, Y_2^t, ..., Y_N^t\}_{t=T+1}^{T+L}, \tag{2.2}$$

where the size of the look-back window is of length T, X_i^t is the value of the variate i_{th} in the t_{th} timestep. In the multivariate case M is the number of dimensions of the input series and N is the number of dimensions of the prediction. We get T x $M \rightarrow L$ x N as the sizes for the past and future value matrices.

In one-step forecasting, we want to make forecasts for one future time step only, so

it can be used in short-term forecasting. In multi-step forecasting the future forecasts are made for several future time steps. The multistep forecasting task can be divided further into iterated multistep forecasting and direct multistep forecasting. In the iterated version, the forecast for each time step is dependent on the previous forecasts, and in the direct version, the model directly generates forecasts for each future time step [24].

In the real world, especially in retail demand forecasting, most of the cases in this study are multivariate time-series forecasting problems. In the context of time series prediction for far in the future, the goal is to have the prediction length L as long as possible [37].

In time series forecasting, the main components of the time series are trend, seasonality, cyclical patterns, and noise. The trend shows the overall long-term direction of the series, seasonality takes into account the periodical changes in the series, cyclical patterns are short-term changes, and noise is the result of stochastic fluctuations such as measurement errors [24].

The **Big O notation** is used when comparing the complexity of the different models. It describes the upper limit of the complexity of the algorithm, showing the impact on time and memory usage as the length of the series T increases.

2.2 Formulas for loss functions

To evaluate the accuracy of the forecasts, several metrics are used in the articles that support this study, defined as follows.

Mean Squared Error (MSE) is probably the most popular metric to evaluate the difference between the prediction and the actual value. MSE sums up the squared differences of each predicted and actual value pair. The formula is as follows:

$$MSE = \frac{1}{n}\Sigma_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{2.3}$$

where y_i stands for the actual values, $\hat{y}_i$ stands for the predicted values, and n is the number of predictions. The same notation is used in all the following formulas.

Mean Absolute Error (MAE) is another commonly used metric, which calculates the absolute error between the prediction and the actual value for each sample, then sums up and divides the result by the number of samples. The direction of the errors is not considered. The formula is as follows:

$$MAE = \frac{1}{n}\Sigma_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.4}$$

.

Mean Absolute Scaled Error (MASE) is defined as follows:

$$MASE = \frac{MAE}{\frac{1}{n-1}\sum_{i=2}^{n}|y_i - \hat{y}_{i-1}|} \tag{2.5}$$

.

Root Mean Squared Error (RMSE) is the root of the mean square error. It can provide information about the magnitude of the deviations.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2.6}$$

.

Mean absolute percentage error (MAPE) expresses the accuracy of the prediction as a percentage. The formula is as follows:

$$\text{MAPE} = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{2.7}$$

.

Weighted absolute percentage error (WAPE) is defined as follows:

$$\text{WAPE} = \frac{\sum_{n=1}^{N}\sum_{t=1}^{T}|y_{n,t} - \hat{y}_{n,t}|}{\sum_{n=1}^{N}\sum_{t=1}^{T}|y_{n,t}|} \tag{2.8}$$

.

Weighted Quantile Loss (WQL) evaluates how well a probabilistic model predicts various quantiles of the future distribution. It is used for models that instead of point forecasts try to predict probability distributions. The formula is as follows:

$$\text{WQL}_{q_j} = \frac{1}{\sum_{i=1}^{N}\sum_{t=T+1}^{T+L}|y_t^i|}\sum_{i=1}^{N}\sum_{t=T+1}^{T+L}\rho_{i,q_j}\left(y_t^i, f_t^{i,q_j}\right) \tag{2.9}$$

,

where N is the number of time series, T is the length of the series i, H is the prediction length. The value of the i-th time series at time t is y_t^i , the predicted quantile q_j of time series i at time t is f_t^{i,q_j}, and $\rho_{i,q_j}(y_t^i, f_t^{i,q_j})$ is the quantile loss at q_j.

2.3 Naive and traditional methods for time series forecasting

The simplest method of all the forecasting techniques used in this text is the **naive method**, which sets all forecasts equal to the value of the newest observed value. A more developed method is the seasonal naive method, **sNaive**, which takes also seasonality into account. For example, when used to predict daily sales for each week,

it uses the most recent value of each weekday to produce the forecast for the same weekday in the future. [41].

Perhaps the simplest actual methods used to make product-level forecasts are regressive univariate forecasting methods. These models use only past sales history to make forecasts for the future. Some examples are **ARIMA**, **exponential smoothing**, Fourier analysis, and state space models. In some higher aggregate-level forecasts, these simple methods can still perform well enough [12]. Since traditional methods are simple and well studied, they can also be used as a benchmark when evaluating new methods.

ARIMA (AutoRegressive Integrated Moving Average) is a popular statistical method used for time series forecasting. It is a classic method widely applied in various fields of industries. The model is defined by three parameters:

- AR (AutoRegressive) component (p): Represents the relationship between past values and the current value in a time series.
- I (Integrated) component (d): Represents the number of differences required to make the series stationary.
- MA (Moving Average) component (q): Represents the dependency between an observation and a residual error from a moving average model.

The general ARIMA model is denoted as ARIMA(p,d,q), where p indicates the number of lag observations in the model, d represents the number of times differences are applied to achieve stationarity, and q refers to the size of the moving average window. The advantages of an ARIMA model are: it is suitable for short-term forecasting when enough historical data is available, it works well with univariate time-series data, and it can model both trend and seasonality. Some of the limitations are that it assumes a linear relationship in the data, requires a careful selection of the parameters, and may struggle with highly volatile complex patterns.

Exponential smoothing is another classic method for time series forecasting. Exponential smoothing assigns exponentially decreasing weights to past observations, which means that recent data points have more influence on the forecast than older ones. This makes it highly effective for capturing trends while smoothing out random fluctuations. The three main types of exponential smoothing are: simple exponential smoothing (SES), which is suitable for time series data without trends or seasonality; Holt's linear trend model (Double Exponential Smoothing), which adds a trend component to SES; and Holt-Winters method (Triple Exponential Smoothing), which makes the method suitable also for seasonal time series. The general formula for simple exponential smoothing can be written as follows:

$$S_t = \alpha X_t + (1 - \alpha) S_{t-1} \tag{2.10}$$

where S_t is the smoothed value at time t, X_t is the actual observation at time t, and α is the smoothing factor that determines how much weight is given to the most recent observation. The advantages of exponential smoothing are that it gives more importance to recent data, it is simple to implement, it works well for short-term forecasting with limited data, and it can handle trends and seasonality. However, it is not ideal for long-term forecasting, it does not perform well with sudden changes or external shocks, and it requires fine-tuning of smoothing parameters.

2.4 Tree based methods for time series forecasting

Artificial intelligence serves as the overarching field that encompasses various computational techniques that enable machines to exhibit intelligence behavior. ML is a subset of AI that focuses on learning from data to make predictions or decisions. Within machine learning, there exist several paradigms and methodologies. Machine learning (ML) has emerged as a transformative technology, enabling computers to learn patterns and make decisions without explicit programming. Founded in artificial intelligence (AI), machine learning uses statistical techniques to train models on large datasets, facilitating automation and decision making in various domains. Rapid advances in computational power, data availability, and algorithmic development have taken ML to the forefront of technological innovation, impacting industries such as the retail industry, especially time series forecasting methods used in demand forecasting. Classical machine learning methods include decision trees, Support Vector Machines, and Random Forests.

Tree-based methods are one of the first machine learning methods used to solve time series forecasting problems. They have several advantages compared to traditional statistical methods and with the latest deep learning methods. They outperform traditional methods in accuracy and are more user-friendly than black-box methods. Tree-based methods are particularly useful when the data exhibit non-linearity that traditional models struggle to capture, there are multiple influencing factors beyond time-based trends, when there is a need for fast or scalable solutions, or when the data contain missing values or outliers.

Decision trees are one of the most fundamental machine learning algorithms, widely used for classification and regression tasks. In time series forecasting, decision trees provide a powerful alternative to traditional statistical models by capturing nonlinear relationships and handling multiple influencing factors.

A decision tree is a hierarchical model that divides the data into subsets based on decision rules. It consists of nodes, branches, and leaves. The tree grows by recursively partitioning data based on the most significant features, making it easy to interpret

and visualize.

Unlike traditional time-series models like ARIMA, decision trees do not explicitly model time dependencies. Instead, they rely on feature engineering to extract useful patterns from historical data. The key steps of decision tree models include:

1. Feature engineering: lag features, rolling statistics, time-based features, and external variables.

2. Training the model: the decision tree splits the data at each node based on selected features to minimize the prediction error. For regression tasks, the mean squared error or mean absolute error is used to find the best split.

3. Making predictions: once trained, the decision tree uses historical patterns to predict future values.

The advantages of decision trees for time series forecasting are that they can handle nonlinearity, require minimal preprocessing, handle missing data well, and are easy to interpret and explain. This last point makes decision trees very user-friendly. Disadvantages are that they easily overfit, they have limited time dependency awareness unless engineered as features, and they require large datasets. Decision trees also serve as a strong foundation for ensemble models such as random forest and XGBoost, which significantly improve prediction accuracy.

Random Forest is a powerful ensemble learning method that improves prediction accuracy and robustness over traditional decision trees. It is an extension of decision trees that combines multiple trees to create a more accurate and stable prediction model. A Random Forest builds multiple decision trees on different subsets of the data and then averages the predictions. Random forests reduce overfitting compared to a single decision tree.

Gradient Boosting Machines (GBM) build strong predictive models by sequentially combining multiple weak models, like decision trees, optimizing performance through iterative learning. A GBM constructs multiple decision trees sequentially, where each new tree corrects the errors of the previous ones. Unlike Random Forests, GBM learns from mistakes iteratively, improving performance at each step. Some powerful GBM implementations are XGBoost, LightGBM, and CatBoost.

2.5 Deep learning methods for time series forecasting

In the past years deep learning methods have emerged as a powerful alternative for the traditional methods, being capable of capturing complex temporal dependencies

and patterns in large datasets. Deep learning models excel at processing sequential data and uncovering intricate relationships in time series. Some of the key advantages include the ability to model non-linear and complex dependencies, automatic feature extraction, handling of long-term dependencies, and scalability. Deep learning (DL) is a subset of ML that uses Artificial Neural Networks (ANN) to model complex patterns in large datasets.

Recurrent neural networks (RNN) are a class of deep learning models specifically designed for sequential data. Unlike traditional neural networks, which treat each input independently, RNNs retain memory of previous inputs, making them highly suitable for time series forecasting. The core of an RNN is a hidden state that stores information from previous time steps. At each time step, the network receives an input, the hidden state is updated based on the new input and the previous hidden state, and the output is generated and passed to the next step. This process allows the model to capture sequential dependencies and trends over time. The advantages of a RNN are that it captures temporal patterns, it works with variable-length sequencies, it handles nonlinearity, and it learns without extensive feature engineering. Some of the problems of RNNs are that gradients tend to vanish, making it difficult to learn long-term dependencies, they are computationally expensive, and standard RNNs struggle with remembering information over long sequencies.

Convolutional neural networks (CNN) are also one type of deep learning model. Through the use of convolution kernels, it tries to tackle some problems like the vanishing gradient problem. Using convolutional layers decreases the number of parameters needed, making CNN more efficient than a fully connected deep neural network.

Long Short-Term memory (LSTM) networks have emerged as one of the most powerful deep learning architectures for time series forecasting. They excel at capturing long-term dependencies in sequential data, making them ideal for time-series forecasting applications. LSTM is a type of RNN that introduces a memory cell to retain important information over long sequences. Unlike standard RNNs, which struggle with the vanishing gradient problem, LSTMs can effectively learn and store dependencies over extended time horizons.

LSTMs introduce gates that control the flow of information through the network: the forget gate to determine which information should be discarded from memory, the input gate to decide what new information should be stored, and the output gate to control the final output based on the current state of memory. These gates allow LSTMs to selectively retain relevant past information while forgetting unnecessary details, making them effective for time-dependent forecasting tasks. The advantages of LSTMs include that they can capture long-term dependencies, handle nonlinear

patterns, work with multivariate time series, and do not require extensive feature engineering. However, they are computationally very expensive, they are sensitive to hyperparameters, they require large datasets, and they are hard to interpret.

Gated recurrent units (GRU) are a type of recurrent neural network architecture designed to handle sequential data efficiently. They are particularly useful in time series forecasting, where capturing long-term dependencies and trends is crucial. They offer a simplified but effective alternative to LSTM networks.

A **GRU** is an advanced variant of RNN that uses gating mechanisms to regulate the flow of information within the network. These gates help to overcome the vanishing gradient problem, allowing GRUs to capture long-term dependencies more effectively than standard RNNs. Unlike LSTMs, which use three gates, GRUs simplify the structure by using only two gates: the reset gate determines how much past information should be forgotten and the update gate decides how much new information should be passed to the next time step. These mechanisms allow GRUs to efficiently retain or discard information, improving performance while reducing computational complexity. GRUs can capture long-term dependencies without excessive computational cost, handle missing or irregular data better than traditional models, and learn complex temporal patterns without requiring extensive feature engineering. They are less powerful than LSTMs for highly complex patterns, require hyperparameter tuning, and are difficult to interpret.

A **temporal convolutional network (TCN)** is a fully convolutional neural network designed to process sequential data while maintaining a temporal structure. Unlike RNNs, which rely on recurrence, TCNs use causal convolutions to ensure that predictions are made only using past information, making them ideal for time-series forecasting. Some of the key features of temporal convolutional networks are causal convolutions to ensure that future data points are not used for predictions, residual connections to help prevent vanishing gradients and to improve training stability, and parallel computation to allow for the processing of entire sequences in parallel, leading to faster training. The advantages of TCNs are that they capture long-term dependencies, they are highly parallelizable, they are more stable to train, and they have lower memory requirements. The disadvantages are that they are not optimal for short sequences, they require careful tuning, and they have a more complex architecture.

Prophet is a model originally developed by Facebook. It can handle non-linear trends, seasonality, and holidays very well [6]. The algorithm has three main components: a trend component to model non-periodic changes in the data, a seasonality component, and a holiday component to model the influence of different holidays [6].

DeepAR is a forecasting model based on autoregressive recurrent neural networks. It can take as input several time series and make predictions for them all. In

retail, where various external features affect sales, this is a very useful property. [33]

DLinear is an interesting model, which first decomposes the raw input series into a trend component and a seasonal component, which are then run through two linear components of one layer to get the final prediction [46]. A simple model that has shown pretty good results in some studies.

2.6 Transformer-based models for time series forecasting

Transformers are deep learning architectures originally developed for natural language processing tasks, such as machine translation and text generation. However, their ability to handle long-range dependencies and process sequences in parallel has made them highly effective for time-series forecasting [37]. The self-attention mechanism allows the model to dynamically weigh the importance of different time steps, and the positional encoding provides a sense of order to input data, crucial for time series. Parallel processing of all time steps improves efficiency, and Transformers are highly scalable to handle long time series efficiently without recurrent connections.

The architecture of a basic transformer can be seen in the given figure (see Fig. 2.1).

The encoder generates a latent representation of the input, and the decoder component uses this representation to produce the output, using the previous outputs as additional inputs [27]. There are two sub-layers in the encoder part, namely a self-attention mechanism and a position-wise feedforward neural network. The decoder layer is similar, but with an additional sublayer to prevent the model from using future data when making predictions [27].

The core components of transformer-based models are the attention mechanisms, which enable the model to focus on selected input segments instead of the entire data. The attention mechanism calculates the relative importance of the individual elements within the given sequence [8]. The computation of the attention involves three components, namely queries Q, keys K, and values V. The self-attention can then be computed as

$$\text{SelfAttention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{D_k}})V. \tag{2.11}$$

where $Q = XW^Q, K = W^K, V = XW^V$, where W is the corresponding learnable weight matrix and X is the input matrix. Since a single self-attention mechanism is not enough to catch parallel relationships, multi-head self-attention layers are used in most cases. [27].

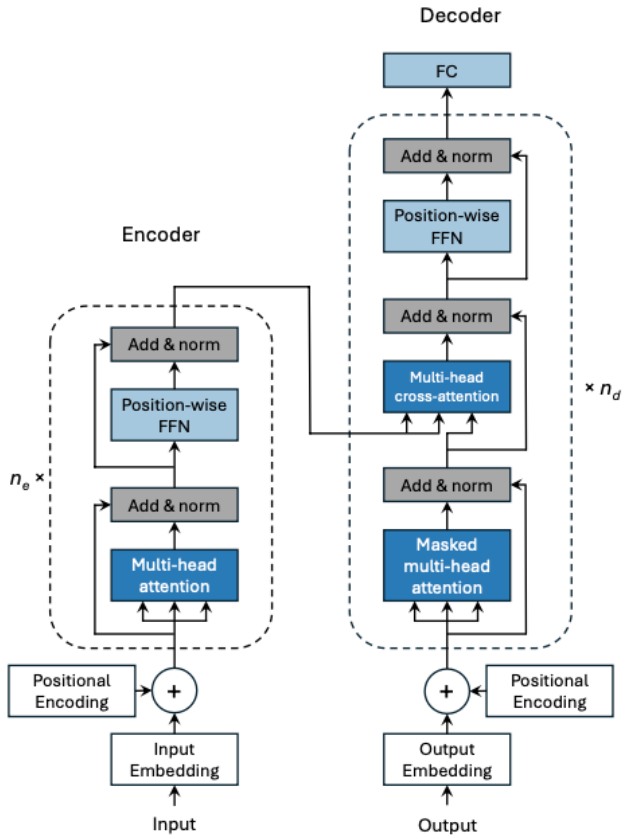

Figure 2.1: A schematic representation of the Vanilla transformer showing the encoder and decoder structure [27].

Another important part of a time series transformer is the positional encoding module, which is needed to keep the time stamps of the data available. Absolute positional encoding is the basic version, where time stamps are coded as is, but recently relative positional encoding and rotary positional encoding have been introduced for better results [1].

When transformers are used in time series forecasting, many transformer-based models use an encoder-decoder structure. The encoder processes the input time series data and extracts meaningful patterns. The decoder generates future predictions based on the encoded information. Some advantages of transformer models are that they handle long-term dependencies effectively, they use parallel processing to speed up training and inference, they can capture complex patterns, and they are scalable to large datasets including multivariate and high-dimensional time series data. As disadvantages, they have a high computational cost, they need large datasets, and they require careful hyperparameter tuning. Since the Vanilla Transformer has many disadvantages, several more effective versions have been developed during the past years.

LogTrans is one of the earliest models introduced already in 2019. It is a model with self-attention convolutional operations, which enables better integration of local

data into the mechanism. The computational complexity is reduced by the logarithmic sparse attention mechanism [37].

One of the first attempts to solve some of the issues with **Vanilla Transformer**s in Time Series forecasting was in 2020 the **Informer** [27]. It uses a ProbSparse self-attention mechanism to be more efficient. The ProbSparse focuses only on critical query-key pairs, which should reduce both the time complexity and the space complexity [27]. It also uses a self-attention distilling mechanism, which iteratively compresses the length of the input sequence, making the model even more efficient [27]. The Informer architecture also includes a generative style decoder that predicts the entire output sequence in a single forward operation, making the process faster and more accurate [27].

The **Informer-Based Time Series Forecasting model (IBTSF**) is a transformer-based model and, therefore, it captures well long-term dependencies. It utilizes temporal attention mechanisms, frequency embeddings, and also external factors embedding, which enables handling of complicated series and missing data values. An important difference compared to the ARIMA and LSTM models is that the IBTSF provides probabilistic forecasts of future values, which is often better than point forecasts[3].

The **Autoformer**, introduced in 2021, is an advanced transformer variant that introduces seasonal trend decomposition within the attention mechanism, enhancing performance on periodic time series data. It is built on the standard encoder-decoder architecture, but with an additional series decomposition block, which separates trend-cyclical components and seasonal components [27]. The **AD-autoformer** is an improved version of the standard Autoformer introduced later [8].

Pyraformer, introduced in 2021, is one more model developed to decrease the complexity of Transformer- based models using sparsity. It uses a pyramidal attention module to describe the temporal dependencies in the observed time series [23]. The pyramidal graph consists of two parts, one for inter-scale connections and one for intra-scale connections [23]. The Pyramidal Attention Module of Pyraformer can effectively extract and integrate features at different time resolutions [23].

The **Temporal Fusion Transformer (TFT)** was introduced in 2021. The TFT selection networks identify and select the most relevant input variables at each time step, which greatly reduces noise and improves accuracy. It has specialized encoders for static covariates, a combination of sequence-to-sequence layers and multihead attention mechanisms, and it can produce prediction intervals instead of only point forecasts [27]. For multi-horizon forecasting the TFT model introduces several improvements compared to basic RNN's and also earlier Transformer-based models. It uses static covariate encoders and gating mechanisms, it has a sequence-to-sequence layer to pro-

cess known and observed inputs, and It has a temporal self-attention decoder to learn long-term dependencies [20]. The most relevant input features are selected by variable selection, the Gated Residual Network blocks increase efficiency, and time-dependent processing is based on LSTM and multihead attention [20].

In 2022 the **Fourier Enhanced Decomposition Transformer (FEDformer)** was introduced. It integrates the Fourier transform into the attention mechanism to efficiently capture frequency components in time series data [37]. It has a periodic trend decomposition, where Fourier transform is used to convert from the time domain to the frequency domain [17]. The main modules of a FEDformer are the frequency enhancement module, the trend decomposition module, and the frequency enhancement attention module [17].

The **Patch time series transformer (PatchTST)** was introduced in 2023. It is designed for multivariate time series forecasting. It uses segmentation of input data into subseries-level patches and channel-independent processing. The patches serve as input for the Transformer [27]. The PatchTST is built on the Transformer encoder, enabling it to capture long-range dependencies [44].

The **CLformer** was introduced in 2023. There are two main differences in this model compared to other transformer-based models. First, it uses a Temporal Convolution Block (TCB), where different weights are assigned to different time steps to capture the local temporal dynamics, and the convolution operations are able to refine repeating patterns [39]. Second, it uses a locally grouped autocorrelation (LGAC), which enables the model to better capture temporal patterns on multiple scales[39].

Another model combining the time and frequency domains is the **Joint Time-Frequency Domain Transformer (JTFT)**, introduced in 2024. The JTFT tries to combine the best parts of time and frequency domain representations of the input series [9]. It can handle both multi-scale dependencies and local relationships, doing this in linear computational complexity [9]. The main pillars of the JTFT are a customized discrete cosine transform module to capture components of the frequency domain (FD), a representation of time series with FD and the latest time domain components, and a low-rank attention layer to learn cross-channel interactions [9].

The **NPformer** was introduced in 2024 [38]. It uses a multiscale segmented Fourier transform on different segments to find frequency-domain correlations between the segments.

The **AD-Autoformer** was introduced in 2024 specifically designed for long-term time series prediction [8]. It uses position embedding to improve the understanding of complicated patterns and relationships in data and a self-attention distilling mechanism to improve performance and generalizability [8].

In 2024 also the **iTransformer** saw daylight [25]. It uses the basic components of

an encoder-only transformer architecture, but in a new way. Attention and feedforward networks are applied on the inverted dimensions [25]. By embedding the entire time series of each variate, instead of each variate of the same time step, multivariate correlations should be modeled in a more efficient way [25]. The iTransformer is actually not a single model, but a new way of thinking about the Transformers in time-series forecasting, and it can be applied to several Transformer-based models.

The **GRAformer** was introduced in 2024. Adding a gated residual attention unit to the transformer-based model is one solution introduced to maintain accuracy while decreasing the complexity of the model [42]. The GRAformer also introduces channel embeddings, which tries to tackle the disadvantages of handling multivariate time series as separate univariate sequences, as many other solutions do [42].

In the **RSMformer**, introduced in 2024, the original input sequence is first encoded by the embedding layer, in which the position, the original sequence, and a global time stamp information are fused. The output is then used as the input to the encoder and decoder. In this model, the standard self-attention mechanism is replaced by a residual sparse attention mechanism to reduce the time- and space-complexity[37].

Another model from 2024, the **TS-Fastformer**, introduces some interesting features to tackle both accuracy and complexity problems. First, it uses a sub-window tokenizer (SWT) to reduce the time complexity of a self-attention operation used in the vanilla transformer based methods. SWT reduces the input length by using averaged windows of the input data [16]. Second, the TS-Fastformer uses a Time-series Pretrained Encoder (TPE) to reduce the effects of a vanilla transformers encoder bottleneck. It is trained using both the added timestamp data and the original data. The use of TPE is fast to train and achieves a good level of performance [16]. Third, it uses a Past Attention decoder to tackle the problem of vanilla transformers, where the inference can only be done in a step-by-step manner and the training with one forward operation. The Past Attention Decoder captures long-term dependencies from the whole input data and short-term dependencies from the most recent sequences of the data [16]. The TS-Fastformer can be used with or without the sub-window tokenizer, depending on the use case.

The **Autoregressive Multimodal Transformer** was introduced in 2024. This model uses an autoregressive architecture with a masked self-attention mechanism, which iteratively generates the predictions of future time steps, conditioned on previously predicted data. It can also be used for forecasts with minimal or no historical data using exogenous data, for example, different types of product-related attributes [31]. The model uses three embedders to extract attribute data, namely the image embedder, the text embedder, and the temporal embedder [31]. The zero-shot approach makes this model good in predicting sales for new products without historical data

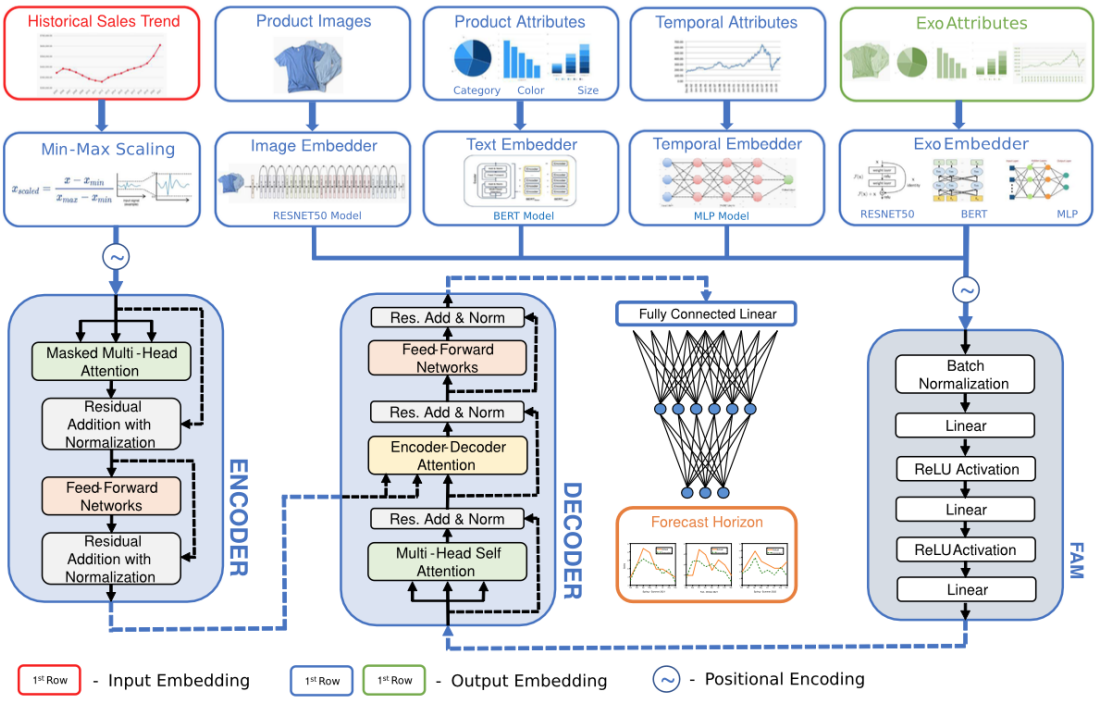

Figure 2.2: A representation of the Autoregressive Multimodal Transformer architecture with three main components: an encoder for processing historical data, a decoder for generating sales forecasts, and a feature aggregation module for fusing the positional encoded vectors [31].

[31]. The architecture of the Autoregressive Multimodal Transformer can be seen in the given figure (2.2). It shows that the structure is a lot more complicated than in the Vanilla Transformer illustrated in figure (2.1).

One of the most advanced Transformer-based models is the **Temporal Feature Enhanced Transformer (TFEformer)**, introduced in the end of 2024 [45]. It has a multilayer encoder architecture, and the complex decoder structure of traditional transformers is replaced by a linear layer [45]. The multivariate input is fed through a multiscale patch embedding and a series-wise embedding to the patch series attention mechanism [45]. A gated feedforward network is used to extract temporal features [45].

In 2025 the **Multiresolution Transformer (MR-Transformer)** was introduced [50]. The MR-Transformer consists of two main modules, the Transformer module with an encoder-decoder structure, and a temporal convolution module [50]. The multivariate series is fed into the Transformer, and the time series of each individual variable is fed into the temporal convolution module [50]. The results are then combined to get the final prediction.

Foundation models are a subset of deep learning models that are pre-trained on a huge amount of data, so they can be applied to a variety of problems [19]. Foundation models are often based on Transformer- structure, so in that sense they are not a new

type of model, but the difference lies in the pre-training process. In theory, they do not need to be trained again on a specific data set but can be used as is. This property makes them easy and fast to use. An example of time-series foundation models is the **General Time Transformer (GTT)** [11].

In many cases, models that can capture both local patterns and long-term dependencies are required. Although individual deep learning models such as CNN and LSTM offer distinct advantages, **hybrid models** that combine them have emerged as a powerful approach. A CNN-LSTM hybrid model combines the CNNs ability to detect local spatial or temporal patterns with LSTM excelling to handle sequential dependencies and long-term trends. Transformers are focusing on temporal dependencies dynamically. These kinds of combination can significantly reduce overfitting and increase prediction accuracy. One very powerful hybrid model is the **TCN-Transformer**, which combines the Temporal Convolutional Network with a Transformer model [30] and another the **Hybrid Attention-based Long Short-Term Memory Network (HA-LSTM)** [48], introduced in 2025.

3. Demand forecasting in retail using time-series methods

As already mentioned in Chapter I, accurate and scalable forecasting has always been an important part of decision-making processes at various levels of retail business. At the strategic level, accurate demand forecasting is needed for site selection, market selection, and pricing strategies [12], and the time horizon is often long-term. Assortment decisions, shelf-space decisions, categories to be offered in each store or channel, pricing decisions, and promotional activities, are examples of tactical level decisions where accurate demand forecasts are needed [12] and the time horizon is mid-term. The operational level deals with inventory and supply chain management on a daily basis, and short-term forecasts are needed for decision making [12]. There is no shortage of data or different kinds of forecasting model, but the challenge today is what model would be best. The retail environment is very complicated, including factors such as fluctuating customer demand, seasonal variations, macroeconomic situation, and promotions, just to name a few that have an impact on product demand. In addition to this, each category of products is different.

In retail, sales data are collected at the receipt level, but often different levels of aggregate are needed for decision-making processes. One way of looking at the different aggregate levels is the multidimensional hierarchy given by Fildes, Ma, and Kolassa [12]. The three dimensions given are product, supply chain, and time. The product dimension looks at different levels of product hierarchy, from the total market to each individual stock keeping unit (SKU). The company, chain, distribution center, and store are typical levels of aggregates in the supply chain dimension, and the aggregates of the time dimension refer to the aggregates of the time level. The customer dimension could be thought of as a fourth dimension, especially if the retailer is active in e-Commerce or has a good customer database. Typically, demand forecasts are made at the SKU-store-day level and the results are then aggregated for each use case afterward. In this way decisions at different levels and dimensions are based on the same forecasts and are in balance.

One thing that makes demand forecasting in retail so difficult is the fact that each

decision has an impact on the demand. Changing the price of a product increases or decreases demand [13, 49] and is a key element in profitable retailing [21], markdown [43], and promotions [28], being special cases of pricing decisions. Different kinds of cross-elasticities, such as the cannibalization effect, in the demand for different products should also be taken into account [16]. Space decisions [4] and displays in stores [15] can also have a huge impact on product demand. Another factor is the external variables that affect demand, such as the general economic situation, competitor activities, holidays, and weather. Finally, the amount of data to handle is huge. A typical retailer has hundreds, or even thousands, of stores, and each store can have tens of thousands of products in its active assortment. So, the item-times-store combinations to be counted for each day is massive.

Retailers have access to various exogenous data that can be used to improve the accuracy of demand forecasting. Data such as local store information, economic data, customer data, or special calendar days can be used if the chosen model is able to handle multivariate time-series data as input. With additional feature engineering, information such as promotions and price changes can also be used [41]. Although classical statistical methods are good at handling small data sets that are not noisy or have a strong trend, they struggle with larger and more complicated time series data [24].

Using explanatory variables on the side of pure sales data time series can drastically improve the performance of a forecasting model [7]. Classical methods like ARIMA and exponential smoothing have been widely used for demand forecasting but lack the possibility for multivariate forecasting. On the other hand, the performance of tree-based methods can be improved by using explanatory variables and additional feature engineering [41]. Another study by [32] shows that RNN based models like DeepAR improve their accuracy when adding exogenous variables to the input data. Since retailers have a lot of data available, it is important to find a model that can take full advantage of that data.

Modern transformer-based models can easily handle multivariate time series data as input and therefore can use various kinds of exogenous data to improve the forecasting accuracy. External variables such as promotions and holidays can be incorporated into the model as shown, for example in [30].

In addition, every product category behaves in a different way. Sales of some categories are very stable and easy to predict with almost any model. Some products fluctuate heavily with seasons and some are highly price sensitive. Categories such as fashion often have a very short life cycle and a limited amount of sales history is available. Seasonal products are only sold in one season of the year and need a special kind of treatment. Campaigns and markdown periods add to the list of things to be

taken into account when making sales predictions in retail.

A good model for retail forecasting should be able to make an accurate multistep prediction for the short- and long-term horizon. It should be able to take multivariate time series as input and have minimum complexity in time- and space- usage. A good model can also use different types of data source, with fast training and prediction times. Interpretability is also a huge advantage in real-world applications. In this chapter, I will discuss the advantages and disadvantages of several available models to find out which kind of model would best meet the requirements of today. Since we focus on forecasting needs at strategic and tactical levels, it is crucial to assess the different models based on their ability to make multistep predictions on long-term horizons. When referring to the definitions of time series forecasting in Equation (2.1) for univariate time series forecasting and in Equation (2.2) for multivariate time series forecasting, this means large values of L.

3.1 General performance of different models in time series forecasting

Traditional statistical methods, such as ARIMA and Exponential Smoothing, cannot capture complex relationships in retail sales data. These models assume a linear relationship over time and are incapable of modeling long-term dependencies [50, 24]. Traditional methods have been widely used and still perform pretty well for simple linear time series with long historical series available [48, 36]. There are several recent studies [48] that show that the precision of these traditional methods cannot compete with more modern methods, even when the precision is enhanced by incorporating additional data, such as promotional information [36].

Tree-based methods have several advantages: they are often more accurate than traditional methods, they are easy to implement and use compared to neural networks, and they have a good level of interpretability [41], which makes them often preferable among real-life users, who want to know what is affecting predictions.

As mentioned in the previous chapter, among deep learning algorithms, recurrent neural networks have long been a popular choice for time series prediction, although they have several limitations [27]. One of the problems is due to the way they are trained, namely backpropagation and gradient descent. In recurrent neural networks, as in many deep neural networks, backpropagation may cause vanishing or exploding gradients [27, 24]. This becomes a problem as the series get longer, which is the case, for example, in retail sales data. RNNs also have the property that in long series they cumulate the error, which is one more disadvantage for them in long-term

forecasting [37]. LSTM and Gated Recurrent Units (GRU) are examples of techniques to help solve gradient-related problems, but even with these improvements, RNNs have problems catching long-term relationships in real-world applications of time series data [27, 24]. When LSTM was tested on several publicly available datasets (weather, stock market, traffic), it was able to catch seasonalities of the data very well, but was beaten in accuracy by the most modern Transformer-based models [3].

Transformers, opposite to RNNs, are capable of using parallelized computation to take full advantage of the parallel processing capabilities offered by modern computers, which brings significant improvements in training speed [27], but vanilla transformer models come with some limitations: they have problems catching local dependencies, they need a lot of memory, and their computational cost is very high [27]. The self-attention mechanism of a Transformer captures long-term dependencies of a time series well, it has a quadratic time complexity, and it also includes elements which are irrelevant and may decrease the accuracy of the predictions [37]. To address these problems, several more sophisticated variants of Transformers have been developed.

One Transformer based model with decent accuracy and complexity is the Patch Time Series Transformer (PatchTST), which has several advantages compared to traditional methods and Vanilla Transformer. It can handle long-term dependencies and is computationally much more efficient than basic transformers. In addition, it is easily scalable and due to its modularity it is easy to integrate new data channels into the model [27] .

In general, the architecture of Transformers allows them to catch very complex dependencies in the data, and as the prediction horizon increases, Transformer- based models show their strength by making more accurate predictions than ARIMA and exponential smoothing, with TFT, Informer, and Transformer showing the best results in [27]. The Informer uses an architecture, which is computationally more effective than the Vanilla Transformer due to three properties, namely the ProbSparse self-attention mechanism, self-attention distilling, and prediction of the entire output sequence in a single forward operation [27]. The Temporal Fusion transformer is efficient and has an advantage in producing more easily interpretable forecasts, which is a huge advantage [20]. As an example, it can provide information about the contribution of the different input variables to the predictions, it can handle several types of input data, it can produce multihorizon forecasts, and it can produce prediction intervals to give the user insight into the range of probable prediction values [27]. In retail, there is a need for several exogenous information, such as location and type of stores, upcoming holidays, and knowledge of the environment. This information can be used in the TFT- model, which is very useful [27]. In tests with publicly available retail data, the Temporal Fusion Transformer has shown very good accuracy results compared to several other

common models [20]. CLformer also belongs to the best performing models among Transformers, which has been shown in multiple studies [39]. It seems to capture local temporal dynamics and seasonal patterns very well, and outperforms another top Transformer, namely Autoformer, in accuracy [39]. One more advantage of CLformer is its memory usage, which is very efficient as the sequence length increases [39].

Tests with iTransformer [25] on publicly available datasets (ECL, ETT, Exchange, Traffic, Weather) against several state-of-the-art Transformer-based models show that in many cases iTransformer can beat the competitors. The iTransformer beats Autoformer and Fedformer in each case and is beaten by DLinear in one case. PatchTST was the second best performer. Another interesting point is that when the inverted modeling was used with known Transformer-based models (Transformer, Reformer, Informer), their performance increased in all cases [25].

One of the recent variants of Transformer-based models is the TFEformer, which has shown very good results in studies with several datasets, unfortunately excluding retail data. It combines patch-wise tokenization and series-wise tokenization to construct insightful temporal information from the input series [45]. This structure makes it very powerful for finding both local and global dependencies in the data. When tested against several other top performing Transformer-based models, including iTransformer, FedFormer, and PatchTST, and a couple of other model types, it achieved first place in most cases [45]. The performance of TFEformer is especially good in high-dimensional feature forecasting tasks [45]. The training time of TFEformer was slightly longer compared to PatchTST and iFormer, but the prediction error was smaller and the memory usage was the same level for the three Transformer-based models [45].

Recent results with the MR-Transformer [50] show such excellent accuracy results that it is worth looking at the model structure and performance a little deeper. The model is designed for multivariate time series (MTS), and the aim is to model the forecasted series for both temporal and variable resolutions [50]. The model has two modules, one for variable-consistent patterns and one for variable-specific patterns [50]. The representations of both modules are then aggregated for the final predictions [50]. An important part of the model is the long-short-term attention mechanism, which captures both long-term and short-term dependencies. Long-term attention uses a vanilla attention mechanism, and an adaptive segment attention mechanism is designed to capture short-term patterns [50]. The variable-consistent module is a Transformer with embedding layer, encoder, and decoder parts, the variable-specific module has a temporal convolution structure [50]. The model was evaluated on 13 different publicly available multivariate datasets, namely ETTh1, ETTh2, ETTm1, Traffic, Solar-Energy, Electricity, ExchangeRate, SMLm1, AMLm2, Appliance, Ya-

hoo, SCITOS G5, and DSIM [50]. MSE and MAE were used as metrics. The other Transformer-based methods in the comparison were Informer, Transformer, LogTrans, and Reformer. The forecast horizon was altered from L=4 to L=720. Alltogether, there were 66 different multistep test arrangements and the MR-Transformer performed best in all of those. In the single-step arrangement the MR-Transformer was the best in 28 out of 40 cases and the second best in the rest of the cases [50].

3.2 Accuracy of Transformer-based models in general

One of the most important features of a good retail sales forecasting model for retail sales is accuracy. Transformer-based models can show a significant improvement in accuracy, compared to other models [27], and in the past few years their use in time series forecasting has been extensively studied.

An early study shows good results achieved by Pyraformer, tested on three datasets and several time horizons. Pyraformer was compared to Informer, LogTrans, Longformer, and Reformer. When measured by MSE and MAE, Pyraformer was the most accurate in all test cases [23]. In another study, the TS-Fastformer shows very good results both in accuracy and efficiency. It can handle complex relationships, long- and short-term, and beats many of the other transformer- based models, including FEDformer, Autoformer, and PatchTST, in tests [16]. The TS-Fastformer shows a reduction of up to 10,1 % in MSE [16].

When AD-Autoformer was compared in five data sets with several other models, it performed exceptionally well [8]. The datasets were load and oil temperature data from power transformers, data of electricity consumption, dataset of daily exchange rates, traffic dataset, and weekly records of disease patients [8]. Measured with MSE and MAE, AD-Autoformer showed the best results in all cases. The forecast horizon varied from L=96 to L=720 and other methods in the study were Autoformer, Informer, LogTrans, Reformer, LSTNet, LSTM, and TCN. The three traditional deep learning models (LSTNet, LSTM, TCN) performed worse than all the Transfromer based models (AD-Autoformer, Autoformer, Informer, LogTrans, Reformer).

The RSMformer is capable of capturing both univariate and multivariate long-sequence time series with excellent prediction results [37]. The RSMformer model has been tested on several univariate datasets with good results, the improvement in MSE is up to 63,59% [37]. The RSMformer also shows good results in multivariate time series forecasting [37]. The data sets in the study were power transformer data, climate data, exchange data, electricity consumption data, traffic data, and patient

Table 3.1: The average results for multivariate long-term forecasts for JTFT, PatchTST, Crossformer, FEDformer, and DLinear. The metric used was MSE and the results are averaged from forecasting horizons of sizes L=96, L=192, L=336, L=720). The best result of each dataset is shown in bold. Table modified from [9].

	JTFT	PatchTST	Crossformer	FEDformer	DLinear
Exchange	**0.289**	0.373	0.723	0.443	0.296
Weather	**0.219**	0.226	0.224	0.310	0.246
Traffic	**0.385**	0.391	0.512	0.604	0.434
Electricity	**0.154**	0.159	0.297	0.207	0.166
ILI	**1.019**	1.480	3.126	2.597	2.169
ETTm2	**0.247**	0.256	0.881	0.291	0.267
PEMS04	**0.125**	0.160	0.152	0.328	0.240
PEMS08	**0.130**	0.174	0.249	0.382	0.294

data [37]. The RSMformer performs best on data sets with periodicity, seasonality, and large volumes [37]. When the amount of data is smaller, or when there is an implicit dependence on spatial relationships, the performance of the model is less effective [37]. Other models in this study were also Transformer-based models, namely Waveformer, Informer, LogTrans, Reformer, and TCN.

Compared with seven other methods on multiple datasets, including exchange, weather, traffic, electricity, and ETTm2, the JTFT shows the best results [9]. Among transformer-based methods, JTFT was the fastest and had the lowest memory usage [9]. Of the 60 experimental settings, the JTFT was the most accurate in 54, and ranked second in the remaining 6 [9]. A summary of the study results can be seen in the figure given (see Table 3.1). The average accuracy of JTFT measured with MSE is the best in all cases shown.

It can be concluded that JTFT can handle various kind of time series with remarkable accuracy in future predictions. The result is significant, since there were some of the earlier best Transfomer-based models, such as PatchTST and FEDFormer, among the comparison group. All these have shown good results in earlier studies.

Experiments with several data sets (weather, electricity, ETT datasets) show good results for the GRAformer [42]. In these tests, measured by MSE, the GRAformer outperforms several other transformer-based methods in accuracy, with 13% improvement over FEDformer, 23% improvement over Autoformer, and 76% improvement over Informer [42]. It performed equally well with PatchTST, but with a significant improvement in training time [42]. The results of the study can be seen in the figure given (see Fig. 3.1). For GRAformer, similar to JTFT, the results are significant. It beats the best Transformer-based models, which in earlier studies have beaten most of the tradi-

Models		GRAformer		PatchTST/42		DLinear		FEDformer		Autoformer		Informer		Pyraformer		LogTrans	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	**0.148**	**0.199**	0.152	**0.199**	0.176	0.237	0.238	0.314	0.249	0.329	0.354	0.405	0.896	0.556	0.458	0.490
	192	**0.192**	**0.242**	0.197	0.243	0.220	0.282	0.275	0.329	0.325	0.370	0.419	0.434	0.622	0.624	0.658	0.589
	336	**0.244**	**0.282**	0.249	0.283	0.265	0.319	0.339	0.377	0.351	0.391	0.583	0.543	0.739	0.753	0.797	0.652
	720	0.321	0.335	**0.320**	**0.335**	0.323	0.362	0.389	0.409	0.415	0.426	0.916	0.705	1.004	0.934	0.869	0.675
Electricity	96	**0.130**	0.223	**0.130**	**0.222**	0.140	0.237	0.186	0.302	0.196	0.313	0.304	0.393	0.386	0.449	0.258	0.357
	192	**0.148**	**0.240**	**0.148**	**0.240**	0.153	0.249	0.197	0.311	0.211	0.324	0.327	0.417	0.386	0.443	0.266	0.368
	336	**0.166**	**0.260**	0.167	0.261	0.169	0.267	0.213	0.328	0.214	0.327	0.333	0.422	0.378	0.443	0.280	0.380
	720	0.204	0.295	**0.202**	**0.291**	0.203	0.301	0.233	0.344	0.236	0.342	0.351	0.427	0.376	0.445	0.283	0.376
ETTh1	96	**0.372**	**0.394**	0.379	0.401	0.375	0.399	0.376	0.415	0.435	0.446	0.941	0.769	0.664	0.612	0.878	0.740
	192	**0.406**	**0.422**	0.414	0.429	0.412	0.420	0.423	0.446	0.456	0.457	1.007	0.786	0.790	0.681	1.037	0.824
	336	**0.430**	**0.429**	0.435	0.436	0.439	0.443	0.444	0.462	0.486	0.487	1.038	0.784	0.891	0.738	1.238	0.932
	720	**0.445**	**0.460**	0.446	0.464	0.472	0.490	0.469	0.492	0.515	0.517	1.144	0.857	0.963	0.782	1.135	0.852
ETTh2	96	**0.273**	**0.335**	0.274	0.337	0.289	0.353	0.332	0.374	0.332	0.368	1.549	0.952	0.645	0.597	2.116	1.197
	192	**0.337**	0.381	0.338	**0.376**	0.383	0.418	0.407	0.446	0.426	0.434	3.792	1.542	0.788	0.683	4.315	1.635
	336	**0.362**	0.404	0.363	**0.397**	0.448	0.465	0.400	0.447	0.477	0.479	4.215	1.642	0.907	0.747	1.124	1.604
	720	0.405	0.438	**0.393**	**0.430**	0.605	0.551	0.412	0.469	0.453	0.490	3.656	1.619	0.963	0.783	3.188	1.540
ETTm1	96	**0.285**	**0.340**	0.293	0.346	0.299	0.343	0.326	0.390	0.510	0.492	0.626	0.560	0.543	0.510	0.600	0.546
	192	0.335	**0.368**	**0.333**	0.370	0.335	0.365	0.365	0.415	0.514	0.495	0.725	0.619	0.557	0.537	0.837	0.700
	336	**0.363**	**0.388**	0.369	0.392	0.369	0.386	0.392	0.425	0.510	0.492	1.005	0.741	0.754	0.655	1.124	0.832
	720	**0.415**	**0.420**	0.416	**0.420**	0.425	0.421	0.446	0.458	0.527	0.493	1.133	0.845	0.908	0.724	1.153	0.820
ETTm2	96	**0.165**	**0.254**	0.166	0.256	0.167	0.260	0.180	0.271	0.205	0.293	0.355	0.462	0.435	0.507	0.768	0.642
	192	**0.220**	**0.293**	0.223	0.296	0.224	0.303	0.252	0.318	0.278	0.336	0.595	0.586	0.730	0.673	0.989	0.757
	336	0.275	**0.328**	**0.274**	0.329	0.281	0.342	0.324	0.364	0.343	0.379	1.270	0.871	1.201	0.845	1.334	0.872
	720	0.364	**0.382**	**0.362**	0.385	0.397	0.421	0.410	0.420	0.414	0.419	3.001	1.267	3.625	1.451	3.048	1.328
Count		36		17		0		0		0		0		0		0	
% improvement (MSE)				1%		8%		13%		23%		76%		65%		76%	

Figure 3.1: Multivariate long-term forecasting results for GRAformer compared to several other methods with MSE and MAE as metrics. On the left the dataset and forecasting horizon are shown. On the bottom "Count" stands for first positions and improvement tells how much better the GRAformer is compared to other models. [42].

tional methods. Implicitly, it can be said that the most advanced Transformer-based models are more accurate than traditional models or older versions.

In addition to different Transformer-based models, Prophet also shows good results compared to traditional statistical models such as ARIMA [24]. The strengths of Prophet are especially in predicting change points and trends, and it performs well in handling large-scale data [24]. In general, though, it can be concluded that different kind of Transformer based models, including hybrid models, are currently performing best. They are also being studied a lot and developing at an enormous speed.

Foundation models for Time-Series are still more or less in the development phase, but some preliminary results about their performance can already be found. When measured with MSE and MAE, the results on several datasets show very promising results. The GTT Zero-Shot model performed second in almost all cases, and the GTT fine-tuned version was the best in almost all cases. From transformer-based methods, PatchTST, Crossformer, and Fedformer were among the comparison models [11].

3.3 Accuracy of Transformer-based models with retail data

In retail, a versatile model is needed with many special needs. Both long-term and short-term predictions are needed, accuracy is a must, and in many cases speed is key, and there is a need for good interpretability and the option to use static future covariates. The Temporal Fusion Transformer (TFT) fulfills many of these needs: it uses a gating mechanism to skip over unused components, it uses variable selection networks to select relevant input variants at each time step, it can take static covariates as input to support prediction, and it uses temporal processing to learn both long- and short-term relationships using both observed and known time-varying inputs [20]. Instead of point forecasts, the Temporal Fusion Transformer predicts intervals through quantile forecasts, which increases interpretability and is often more useful in retail environment [20]. TFT has been tested with real retail data (Favorita) and the results were more than promising [20]. The TFT model was compared with several state-of-the-art methods and quantile losses of P50 and P90 were used as a metric of accuracy. The TFT model outperformed all competing models. A summary of the results can be found in Table (3.2).

Table 3.2: P50 and P90 quantile losses on Favorita retail dataset. Percentages in brackets show the increase in quantile losses for each model compared to TFT. [20].

	TFT	MQRNN	Seq2Seq	MLP	Ridge	ConvTrans	DeepAR
P50	0.354	0.379 (+7%)	0.411 (+16%)	0.532 (+50%)	0.720 (+104%)	0.429 (+21%)	0.574 (+62%)
P90	0.147	0.152 (+3%)	0.157 (+7%)	0.199 (+35%)	0.163 (+10%)	0.192 (+30%)	0.230 (+56%)

Another study [27] compares several transformer-based methods with some of the traditional methods. The traditional methods used as baselines are the naive method, the seasonal naive, exponential smoothing, and the autoregressive integrated moving average, and the transformer-based methods are vanilla Transformer, TFT, Informer, PatchTST, and Autoformer. The data set is the publicly available M5 competition dataset, which consists of hierarchical sales data from several product categories at the product level. Both point forecasts and probabilistic forecasts are evaluated. In addition, several forecast horizons were tested.

The results show that the transformer-based methods outperform the traditional methods in almost each case, Transformer, Informer, and TFT showing the best results in accuracy [27]. The results can be seen in Table (3.3).

Table 3.3: The performance of Transformer-based methods and traditional methods with MASE as the metric. The column headers show the length of the forecasting horizon in time steps and the model name is given on the left. [27].

Model	1-7	8-14	15-21	22-28	1-28
Transformer	0.8154	0.8874	0.8913	0.8806	0.8687
Informer	0.8164	0.8869	0.8901	0.8815	0.8687
TFT	0.8165	0.8867	0.8910	0.8813	0.8689
Autoformer	0.8344	0.9031	0.9080	0.9058	0.8878
PatchTST	0.8531	0.9215	0.9238	0.9138	0.9031
AutoARIMA	0.9741	1.0240	1.0286	1.0245	1.0128
AutoETS	0.9964	1.0619	1.0687	1.0653	1.0481
Seasonal Naive	1.1488	1.2115	1.2125	1.2111	1.1960
Naive	1.2511	1.3109	1.3190	1.3178	1.2997

When measured by WQL as a metric for probabilistic forecasts, the same kind of results can be seen [27], with TFT and Transformer showing the best results. The results with WQL can be seen in Table (3.4). For time steps 1-7, the TFT is the most accurate, with Vanilla Transformer in second place. Vanilla Transformer shows the best results on steps 8-14, 15-21, and 1-28. As can be expected, traditional models (AutoARIMA, AutoETS) are getting worse results compared to the Transformer-based models, and Seasonal Naive- and Naive-models are not even getting close. One interesting thing to note is that the Vanilla Transformer actually performs equally well with the more recent models.

As can be seen, the differences in Table (3.4) are actually very small between the Transformer- based models, AutoARIMA, and AutoETS.

The Hybrid Attention-based Long Short-Term Memory (HA-LSTM) is one of the recent models that combines the advantages of a Transformer and LSTM [48]. It tries to tackle problems such as capturing local and global temporal patterns, handling multiple seasonalities, interpretability, and scalability. It integrates several multi-head self-attention modules with LSTM layers and has a separate interpretability layer to provide information on the relative importance of the different time steps and features in the predictions [48]. The model was tested with a retail dataset with 54 stores, several product categories and 22170 products against several baselines, including statistical methods, machine learning methods, and deep learning methods. To evaluate performance, three metrics were used, namely RMSE, MAE and MAPE [48]. The HA-LSTM model outperformed all other models, showing excellent results. In general, the statistical models showed the poorest results, the machine learning methods were the

Table 3.4: The performance of Transformer-based methods and traditional methods with WQL as the metric. The column headers show the length of the forecasting horizon in time steps [27].

Model	1-7	8-14	15-21	22-28	1-28
Transformer	0.5298	0.5370	0.5477	0.5594	0.5436
Informer	0.5345	0.5410	0.5524	0.5628	0.5477
TFT	0.5294	0.5376	0.5496	0.5582	0.5438
Autoformer	0.5468	0.5508	0.5621	0.5751	0.5587
PatchTST	0.5365	0.5470	0.5602	0.5681	0.5531
AutoARIMA	0.5712	0.5670	0.5788	0.5934	0.5775
AutoETS	0.5649	0.5686	0.5797	0.5975	0.5777
Seasonal Naive	0.8104	0.7750	0.7788	0.8083	0.7926
Naive	0.8773	0.9630	1.0805	1.2385	1.0407

second best, and the deep learning methods showed the best results [48]. The results were consistent among all categories, but the differences were largest in the most difficult ones such as "Seasonal Clothing" and "Holiday Goods" [48]. However, in categories with stable demand, such as "Office Supplies" and "Home Goods", the performance difference between the best and worst models was smaller [48]. The general average results can be seen in Table (3.5). There is also a clear progression from ARIMA to HA-LSTM, which can be seen. Each more sophisticated model beats the previous ones in accuracy.

Table 3.5: Comparison of different models based on RMSE, MAE, and MAPE metrics. HA-LSTM shows best results on all metrics.[48]

Model	RMSE	MAE	MAPE
ARIMA	1.234 ± 0.021	0.891 ± 0.015	$25.6\% \pm 0.4\%$
ETS	1.198 ± 0.018	0.867 ± 0.013	$24.9\% \pm 0.3\%$
XGBoost	1.021 ± 0.012	0.743 ± 0.009	$21.4\% \pm 0.2\%$
LightGBM	0.998 ± 0.010	0.725 ± 0.008	$20.9\% \pm 0.2\%$
LSTM	0.967 ± 0.009	0.702 ± 0.007	$20.2\% \pm 0.2\%$
Transformer	0.943 ± 0.008	0.684 ± 0.006	$19.6\% \pm 0.2\%$
DeepAR	0.929 ± 0.007	0.674 ± 0.005	$19.4\% \pm 0.1\%$
N-BEATS	0.918 ± 0.006	0.666 ± 0.005	$19.2\% \pm 0.1\%$
HA-LSTM	$\mathbf{0.897 \pm 0.005}$	$\mathbf{0.651 \pm 0.004}$	$\mathbf{18.7\% \pm 0.1\%}$

The HA-LSTM model was also tested in different product categories to gain more insight into the differences between the different categories. The results were clear and HA-LSTM outperformed other models in all categories [48]. Such results show that

the model is robust to various types of sales patterns [48]. As already mentioned, the performance gap was largest in categories with high seasonality [48]. When a temporal analysis was performed to better understand performance variations over time, all models showed an increase in RMSE during periods around major holidays and in the beginning of a new season [48], but also in these cases HA-LSTM demonstrated the best results.

Another hybrid model for sales forecasting integrates Temporal Convolutional Networks with Transformer-based attention mechanism [30]. The results with Favorita sales data can be seen in Table (3.6) and for the Superstore sales data set in Table(3.7).

Table 3.6: Performance comparison of various models with a Hybrid Temporal Convolutional Network and Transformer model (Favorita Grocery Sales Forecasting data). The metrics used are MAE, RMSE, and MAPE. The Hybrid Temporal Concolutional Network shows the best results. [30].

Model	MAE	RMSE	MAPE
LSTM	2.50	3.40	8.50%
GRU	2.45	3.35	8.30%
CNN-LSTM	2.30	3.15	7.75%
Seq2Seq	2.40	3.25	8.25%
DeepAR	2.35	3.20	7.95%
N-BEATS	2.25	3.10	7.60%
TFT	2.15	3.00	7.25%
DSSM	2.30	3.18	7.80%
WaveNet	2.20	3.12	7.50%
Att-RNN	2.22	3.14	7.65%
TCN	2.18	3.08	7.45%
TCN with Transformer	**2.01**	**2.81**	**6.85%**

The TCN Transformer model seems to beat the other models in all measured metrics. Compared with the worst-performing method, namely LSTM, the improvements were significant. The results for TCN with Transformer can be seen in bold on the bottom line with values of MAE 2.01, RMSE 2.81, and MAPE 6.85 %. For LSTM the results for these same metrics are MAE 2.50, RMSE 3.40, and MAPE 8.50 %. TFT was one of the best models in previous studies and was also beaten by the TCN Transformer in this study.

The results with the Superstore data are similar to the results on the Favorita data set. The hybrid TCN and Transformer model beats the other models in all cases.

When looking at all the previous results, it seems pretty obvious that Transformer-based models are getting better and better, beating the other models study after study in overall accuracy. There are, though, cases that are not deeply

Table 3.7: Performance comparison of various models with a Hybrid Temporal Convolutional Network and Transformer model (Superstore sales data). The metrics in this table are MAE, RMSE, and MAPE. TCN with Transformer beats the other models measured with all metrics. [30].

Model	MAE	RMSE	MAPE
LSTM	1.40	2.20	6.25%
GRU	1.38	2.15	6.10%
CNN-LSTM	1.28	2.05	5.85%
Seq2Seq	1.32	2.10	5.90%
DeepAR	1.26	2.00	5.70%
N-BEATS	1.20	1.92	5.40%
TFT	1.18	1.90	5.10%
DSSM	1.25	2.02	5.65%
WaveNet	1.22	1.95	5.50%
Att-RNN	1.24	2.00	5.55%
TCN	1.21	1.93	5.45%
TCN with Transformer	**1.15**	**1.85**	**4.90%**

handled in the previous results, so it is good to also take a quick look at some corner cases.

First, predicting sales of products without historical data, novelty products, is a huge challenge. One way to tackle the problem is to use an Autoregressive Multimodal Transformer, where the model can use historical sales data of similar products to make predictions [31]. The model has been tested with good results for fashion products, which are one of the most difficult categories to predict [31]. The disadvantages of the model are that it is very complex and, therefore, introduces significant computational demands [31]. In addition, the model is extremely dependent on the richness of the training data, such as images and attributes at the product level [31]. A different approach can be found in [34], where they first transformed historical sales into demand using several methods to handle stock-outs and then predicted future sales using a Random Forest. Several product-related features were used to increase the accuracy of the predictions, as the demand for similar products can be used for products with very limited historical data, if any. The precision measured with MAPE was 25,346% when tested in fashion products.

Second, different product categories behave differently. One study [10] uses the combination of a GRU and a LightGBM to build a model to predict the sales of short shelf-life products in e-Commerce. The GRU-LightGBM performs best, measured with RMSE, MAE, and MAPE, but there are several weaknesses in this study, which should be further investigated. First, there was only one product that was included, that is,

fresh milk. Second, a very short sequence of good quality history data was selected, and third, there were no modern models in the comparison group (ARIMA, SVR, LSTM). Comparison of GRU-LightGBM with a stand-alone GRU and LightGBM shows that the integrated model performs better than the components alone [10].

A model called RF-XGBoost is suggested for the prediction of sales of short-cycle agricultural products in [18]. It is another hierarchical tree-based method with RF and XGBoost as its hierarchical components [18]. The tests were made so that the results from the RF- layer were used as the feed for all models in the test (XGBoost, LSTM, BPNN, Decision Tree) and the end results were then compared using MAE, MSE, RMSE and MAPE for accuracy. The RF-XGBoost combination was the most accurate, but the marginals were small [18] and there were no modern Transformer-based models in the comparison group. Another study [47] used a combination of a Transformer and a LightGBM to predict drugstore sales, showing very good results compared to actual sales.

Transformer-based models perform well with each type of product category, so they can be adapted regardless of the area in which the retailer operates [27, 48]. Some results from [27] can be seen in Table(3.8).

Table 3.8: Performance Metrics for Forecasting Methods Across Categories. The metrics used are MSE and WQL and the name of the model can be seen on the left. Transformer-based models beat the traditional models among all product categories. [27]

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 NaÃ¯ve	1.1799	1.1889	1.2218	0.7317	1.0869	0.8497
Naive	1.2817	1.2845	1.3326	0.9576	1.4460	1.1711

As the results in Table(3.8) show, the Transformer-based models beat the Baseline models in each category measured both with MASE and with WQL.

Forecasting the demand of fashion products is probably one of the most difficult

tasks in retail forecasting, and therefore it is useful to look at them separately. They have a very short life-cycle, and the amount of historical sales data is limited, if any are available. Other challenges mentioned in [35] are the lack of sufficient data in general, the demand censoring effect caused by stock-outs, uncertainty in exogenous variables, and intermittent demand. In addition, various sizes, colors, and textures add to the complexity of accurate forecasting in these categories. One possible solution could be the use of interrelated product attributes, as suggested in [31]. In their study, they feed an Autoregressive Multimodal Transformer with several attributes, such as product image, textual descriptions, and temporal attributes to increase the accuracy of forecasts [31]. The results are very promising, with accuracy improvements compared to several other methods. Among fashion products, there are also other categories with similar seasonal behavior, which could benefit from the use of the approach described in [31]. Some results can be seen in Table(3.9).

Table 3.9: Performance comparison of multiple baseline models across diverse data modalities with the Autoregressive Multimodal Transformer (ARM Transformer). The table shows how performance measured by MAE and WAPE improves by adding more modalities. [31].

Approach	**Modality**	**MAE**	**WAPE**
Attribute kNN	[Txt]	5.087	36.26
Image kNN	[Img]	5.009	35.70
Embedding kNN	[Img+Txt]	4.857	34.62
Image RNN	[Img]	2.588	27.49
Concat Multimodal RNN	[Img+Txt]	3.069	32.60
Residual Multimodal RNN	[Img+Txt]	3.135	33.60
Cross Attention RNN	[Img+Txt+Tmp]	2.406	25.56
FCN - CNN	[Img+Txt+Tmp+CT]	2.109	22.40
ARM Transformer	[Img+Txt+Tmp+CT+Exo]	1.546	16.42

There are two main observations to be made from the results in Table(3.9). First, the Autoregressive Multimodal Transformer with all modalities available is performing superbly compared to the other models. Second, adding more information through different data increases the accuracy.

Third, predicting sales for markdown seasons is crucial to eliminate seasonal products at as good a price as possible. The problems are similar to those of the cases already mentioned: very little historical data, intermittent demand, censoring effect caused by stockouts, and the price sensitivity of the products. In addition, since the products are often seasonal, their best-selling season is over. Transformer-based models using exogenous data, both static and time-dependent, should be adaptable also in these cases, but no evidence has been found in the studies so far. In [14], a

deep learning method is used to forecast the sales of fashion products in the clearance seasons, and a theoretical demand model is used to transform the past sales history into demand, similar to the method used in [34]. They showed pretty good results, but the process was very tedious.

3.4 Complexity of Transformer-based models

Since the architecture of transformer-based models can be pretty complex, it can be very slow to train them [27], making them unpractical in real-world solutions such as demand forecasting in retail, since the theoretical complexity of a Vanilla Transformer is of quadratic order or written in O-notation, $O(T^2)$. One of the main reasons for the complexity of Transformers is the attention mechanism. There have been many attempts to tackle this problem, and at least the theoretical complexity of the best models has decreased enormously.

As already mentioned, one of the first attempts to address the complexity problem was the Informer [3] architecture with the ProbSparse self-attention mechanism, the self-attention distillation mechanism, and compression of the input sequence [3, 27], achieving the theoretical time and space complexity of $O(TlogT)$. The same level results were also achieved by the Autoformer [27] architecture and its advanced version AD-Autoformer [8] using a decomposition and autocorrelation mechanism. These kinds of improvement are necessary if the models are to be used in real-world applications of long-term forecasting. The CLformer has similarities with the Autoformer, but differs in the way in which the autocorrelation is computed by computing it within smaller segments of the input [39]. The achieved complexity for the CLformer is also $O(TlogT)$. The Multiscale Residual Sparse Attention Model (RSMformer) also achieves a time complexity of $O(TlogT)$ by using a residual sparse attention (RSA) mechanism [37].

One way to reduce complexity, introduced in the Temporal Fusion Transformer [20], is by using gated residual networks (GRN) [27, 20], which calculate the importance of each input. This kind of architecture makes it possible for the model to adaptively skip unnecessary information, which reduces the amount of data to be handled at each stage.

The Patch Time Series Transformer (PatchTST), designed for multivariate time series forecasting, uses segmentation of the series into smaller patches and channel-independent processing [27]. Patches are used as input tokens for the Transformer, which greatly reduces both time and space complexity [27]. The theoretical complexity achieved is $O((T/S)^2)$, where T is the length of the original series and S is the patch length [27]. The channel-independent structure also comes with several advantages, such as scalability and modularity [27].

Two reasons for slowness in many Transformer-based models are the bottleneck incurred by a deep encoder and the step-by-step structure of the decoder, which the TS-Fastformer tries to solve by introducing three new features [16]. The three components are the Sub-Window Tokenizer, which reduces the length of the input series, a pre-trained encoder, and a Past Attention Decoder [16]. First, the Sub-Window Tokenizer (SWT) compresses the length of the input series from T to T/wT, reducing the complexity of self-attention to $O((T/wT)^2)$ similar to the PatchTST [16]. The Sub-Window Tokenizer is a separate part of the structure and therefore can also be used with other Transformer-based models, as shown in [16]. The training times of FEDformer, Autoformer and Informer improved dramatically when used together with SWT [16].

One of the most advanced transformer-based models, the JTFT, achieves a linear complexity $O(T)$ [9]. Instead of directly using the entire input series, the JTFT uses in most operations the time-frequency representation, so the sequence length remains constant [9]. Other examples of Transformer-based models achieving linear time and space complexity are the Pyraformer [23] and the FEDformer [37]. The Pyraformer uses a novel pyramidal attention model to achieve low time and space complexity while simultaneously capturing long-range dependencies [23].

The GRAformer uses several methods to decrease the complexity of the model. First, like in PatchTST, it uses patching of the original series. Second, the gated residual attention unit formulates attention and GLU as a unified layer, which dramatically improves both parameter and computational efficiency [42]. Compared to PatchTST, the GRAformer is faster in both training time and inference time [42]. The longer the look-back window T, the larger the difference.

There is no exact complexity information available for the iTransformer, but some evaluations have been made. The iTransformer is more efficient than other Transformer-based models in datasets with a relatively small number of variables, and at the same level in datasets with more variables, but also in that case it can be trained faster [25].

A summary of the time and space complexity of the selected models can be found in Table (3.10). These are the theoretical complexities, and in reality some of the more complex models can still have a significant cost because of the large coefficients in the expressions [9].

As already mentioned, although there has been a huge improvement in the theoretical complexity of Transformer- based models, it can be claimed that, in reality, they still are pretty complex and unefficient [46]. A comparison by [46] shows some interesting results with look-back window $T = 96$ and forecasting horizon of $L = 720$, where T and L refer to the univariate time series equation(2.1) and multivariate time

Table 3.10: Time and space complexity of selected models. Several modern variates achieve the complexity of $O(T)$, like JTFT, FEDformer, and Pyraformer. For PatchTST, the complexity depends on the patch size S.

Model	Time Complexity	Space Complexity	Ref
JTFT	$O(T)$	$O(T)$	[9]
FEDformer	$O(T)$	$O(T)$	[9]
AD-Autoformer	$O(TlogT)$	$O(TlogT)$	[8]
Autoformer	$O(TlogT)$	$O(TlogT)$	[9]
Informer	$O(TlogT)$	$O(TlogT)$	[9]
Reformer	$O(TlogT)$	$O(TlogT)$	[9]
Pyraformer	$O(T)$	$O(T)$	[23]
RSMformer	$O(TlogT)$	$O(TlogT)$	[37]
TS-Fastformer	$O(TlogT)$	$O(TlogT)$	[16]
LSTM	$O(T)$	$O(T)$	[9]
LogTrans	$O(Tlog^2(T))$	$O(Tlog^2(T))$	[9]
PatchTST	$O(T/S)^2$	$O(T/S)^2$	[9]
Vanilla Transformer	$O(T^2)$	$O(T^2)$	[9]
MR-Transformer	NA	NA	
iTransformer	NA	NA	
GRAformer	NA	NA	

Table 3.11: Comparison of practical efficiency of Transformers (Transformer, Informer, Autoformer, Pyraformer,FEDformer) and DLinear. The metrics are multiply-accumulate operations (MACs), parameter size, time usage, and memory usage. [46]

Method	MACs	Parameter	Time	Memory
DLinear	0.04G	139.7K	0.4ms	687MiB
Transformer	4.03G	13.61M	26.8ms	6091MiB
Informer	3.93G	14.39M	49.3ms	3869MiB
Autoformer	4.41G	14.91M	164.1ms	7607MiB
Pyraformer	0.80G	241.4M	3.4ms	7017MiB
FEDformer	4.41G	20.68M	40.5ms	4143MiB

series equation (2.2) .

As can be seen in Table (3.11), the real-life complexity of transformers in this case is very high compared to DLinear. Another interesting aspect is that many of the more advanced Transformer- based models cannot beat the Vanilla Transformer, although their theoretical complexity is much better [46]. The publicly available electricity data set was used and MACs stand for multiply-accumulate operations [46].

In some cases, it is possible to decrease the complexity by changing the training strategy. When training the iTransformer, it can be done by selecting only part of the variates in each batch for training [25]. Due to inversion, the number of variate channels is flexible, and the model can predict all the variates [25]. Using this method, memory usage can be greatly reduced without increasing prediction errors [25].

As shown in the previous section, the HA-LSTM model performs very well when it comes to accuracy. Since time matters in practice, it is worth taking a look at the training times and memory usage of the different models compared to HA-LSTM. The original data set has 2,9 million sales records. Traditional methods, ARIMA and ETS, have a memory usage of approximately 100MB, while the memory requirements of machine learning methods range from 400-500 MB and the HA-LSTM needs about 1,5 GB [48]. Traditional methods can process 5000 time series per second, machine learning methods 2500-3000 time series per second, and the HA-LSTM model 1000 time series per second [48]. The training time for traditional methods was approximately 4 hours, while the HA-LSTM model needed 4,2 hours, but the HA-LSTM has a huge advantage compared to traditional methods: it needs only 15 minutes for daily data updates, while ARIMA needs to be fully re-trained each time [48]. It is also worth mentioning that the HA-LSTM model is highly scalable to larger data sets and maintains a high level of performance through parallelization in multi-GPU setups [48]. This is crucial in real-world retail applications.

To find out more about computational efficiency, the HA-LSTM model was compared with the other models also on a different set of retail data. It contained 5 years of daily sales data for 50 items sold in 10 stores [48], so it was a slightly smaller data set. A summary of the results can be seen in Table (3.12).

Table 3.12: Comparison of Models based on RMSE, MAE, MAPE, and Training Time. The table shows that the more complcated the model, the more time to tarin it needs. Among these models, ARIMA is the fastest to train and HA-LSTM requires the highest amount of time. [48].

Model	RMSE	MAE	MAPE	Training Time
ARIMA	5.234 ± 0.121	3.891 ± 0.085	$25.6\% \pm 0.4\%$	0.5h
ETS	4.987 ± 0.098	3.657 ± 0.074	$24.2\% \pm 0.3\%$	0.8h
XGBoost	4.521 ± 0.082	3.343 ± 0.061	$21.4\% \pm 0.2\%$	1.2h
LightGBM	4.398 ± 0.075	3.225 ± 0.058	$20.9\% \pm 0.2\%$	1.0h
LSTM	4.267 ± 0.069	3.102 ± 0.058	$20.8\% \pm 0.2\%$	2.5h
Transformer	4.143 ± 0.064	2.984 ± 0.048	$19.7\% \pm 0.2\%$	2.8h
N-BEATS	4.018 ± 0.056	2.866 ± 0.043	$19.2\% \pm 0.1\%$	3.1h
HA-LSTM	$\mathbf{3.897 \pm 0.051}$	$\mathbf{2.751 \pm 0.039}$	$\mathbf{18.7\% \pm 0.1\%}$	3.2h

As expected, the more complicated the model, the more time it will need to train also in this case. The HA-LSTM model seems to be exceptionally good at capturing daily and weekly seasonal patterns, which is needed in retail time series [48].

In [9] the results are similar when comparing the real training time of seven methods. The highest time usage was measured for FEDformer, Autoformer, Informer, and Crossformer. JTFT and PatchTST showed much better results, and DLinear was in a class of its own with very fast results. In addition, the inference time usage of the same models was also measured, with the same kind of results. The measurements were made with different sizes of the look-back window T and target window L to also find out how much their length affects the results. Changing the size of T increased the time usage of all models, as expected, but the order remained the same. Overall, JTFT performed best of the Transformer-based methods in all cases, beaten only by DLinear. But, as already mentioned, DLinear was in a class of its own. It is worth mentioning that the accuracy of DLinear is very poor, as shown earlier in the figure (see Table 3.1) from [9].

The theoretical complexity measured by the O-notation is only one aspect, and therefore it is also useful to look at other kind of complexity between the different models. Traditional statistical models, such as ARIMA, need a lot of parameter tuning and testing before the model is usable. Tree-based models need multiple human decisions about the structure and depth of the tree. Some studies suggest a half-automated process for more efficient feature engineering and data preprocessing [26]. The suggested

process includes an autonomic product clustering, using feature engineering to find out the demand behavior, and then to use these feature-based clusters to automatically select a suitable ML method for each cluster's demand forecasting [26]. This makes the process more complicated but takes advantage of the different behavior of each product cluster. This kind of approach requires building of several ML-models and careful preprocessing of the data, but after set-up it is able to select the best model for each use-case. On the other hand, Transformer- based models are themselves more complicated, but they are very flexible and do not need that much preprocessing after the configuration.

3.5 Interpretability of Transformer-based models

In addition to the accuracy and complexity of the models, usability, especially interpretability, is one of the key elements in real-life solutions for time series forecasting models. Traditional statistical models, such as ARIMA or ETS, are easy to use and understand for end users. The weights of different lags are more or less coded in the model and are easily explained for the user. The implementation process is also pretty straightforward, but needs a lot of work when fine-tuning the parameters and selecting the right model-type. Tree-based models also have a high level of interpretability, if the user is interested in finding out which variables affect the results the most.

Deep learning methods, especially transformer-based models, are different. The basic versions do not offer any insight into the reasons behind the forecasts, and it is not desirable, sometimes even unacceptable, to blindly trust the results of a forecasting model [2]. Therefore, it is crucial to build the trust of users and to ensure the reliability of the results by adding some kind of information about the reasoning behind machine-made results [2]. In addition, new laws have been introduced to tackle some of the problems related to interpretability of machine learning results, and even a new subfield of deep learning, xDNN, has grown to study the field of the interpretability of deep learning methods [2].

Since in retail, users often want more than just a single forecast value, much effort has been put into improving the interpretability of the forecasts made by transformer-based models. One example is the probabilistic forecasts, which, instead of a single value, give also information about the uncertainty of the forecasted values. Most transformer-based models can be used to generate interval-based forecasts, as shown, for example, in [20, 7]. Some examples of how the visualization of confidence intervals can help the user understand the predictions are given in Figure (see Fig. 3.4) from [27]. In the example, an interval of 80% is used, but depending on the use case other intervals can be chosen as well. The shaded area around the predicted line of point forecasts

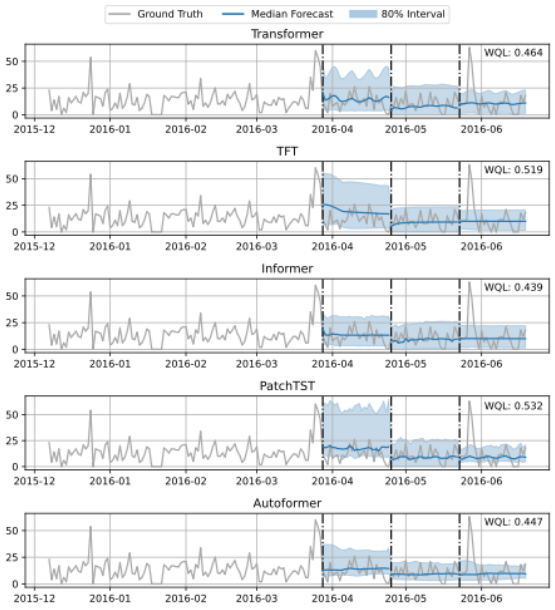

Figure 3.2: Examples of confidence interval visualizations in probabilistic forecasting. The figures show mean values with shaded confidence intervals for the predicted future values for Transformer, TFT, Informer, PatchTST, and Autoformer. [27].

shows the interval of 80% and the narrower the shaded area, the less uncertainty in the prediction. This kind of representation does not give any information about the inference process behind the forecasts, but it gives the user information about the trustfulness of the forecasts: the wider the shaded area, the more uncertainty there is.

Another way to improve the interpretability of the models is to give the user more information about what lies behind the forecasting results or how much weight the model puts on the different variables when making the forecast. An example can be seen in Table (3.13), where the importance of input variables for TFT is shown [20]. The table is divided into three categories, namely static covariates, past inputs, and future inputs. Static covariates are variates that do not change in accordance with time, past inputs are exogenous inputs that have affected the past sales, and future inputs are some known future values for the exogenous variables.

As can be seen, for past input, past sales values are critical. For future inputs, promotional information and holidays play an important role. For static variates, the largest weights are attributed to identifying variables such as item number and store number. With this kind of information on the side of the forecasts, the user can better understand the reasoning behind the predicted values. In addition, it might help improve the model by finding new variables to use in it and otherwise improve the model itself.

As also mentioned multiple times, the use of exogenous variables can greatly increase the accuracy of the forecasting models. When such variables are used, the

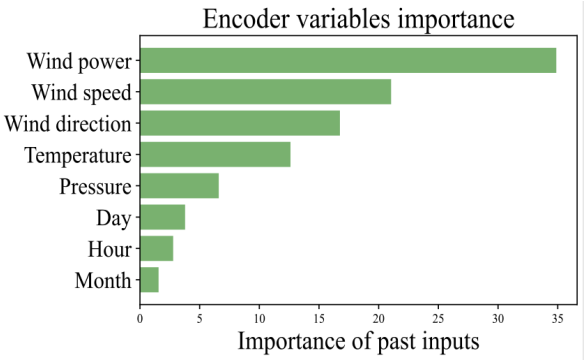

Figure 3.3: Visualisation of variable importance. The vertical axis shows the name of the variables and the horizontal axis the importance. This kind of bar-chart makes it easy for the user to quickly see which variables matter the most. From [22]

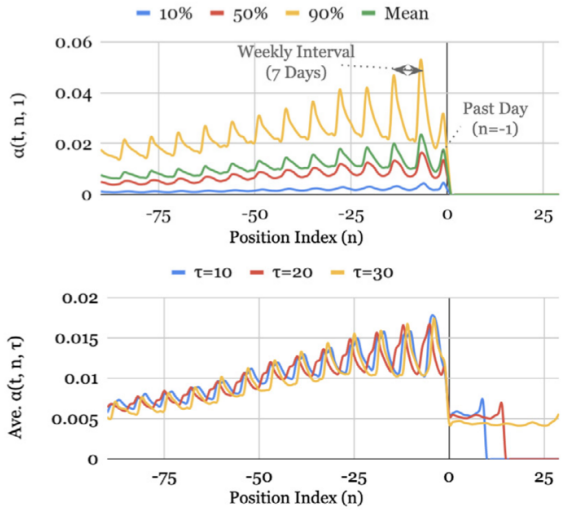

Figure 3.4: Temporal patterns across the retail dataset for TFT. The upper plot shows percentiles of attention weights for one-step-ahead forecasts and the lower plot average attention weights for forecasts at various horizons. From [20].

user is often interested in how each variable affects the final results. Some models offer the possibility to visualize this kind of information, as shown in [22] and in Figure (see Fig. 3.3). Information like this is easy to understand and may increase the trust of the user, as she can understand the reasons behind the forecasts better.

Some of the models can also provide information about the importance of past time steps. The Temporal Fusion Transformer (TFT), which can learn seasonal patterns and lag analysis from raw training data , is an example of that. A visualization of such information is given in Figure (see Fig. 3.4).

Clear weekly patterns can be observed with a trend pattern of decay with the last few days dominating the importance [20]. As mentioned, TFT learns these temporal patterns from raw training data, so there is no need for human hard-coding [20]. This kind of information is expected to be useful in building trust among users and can also

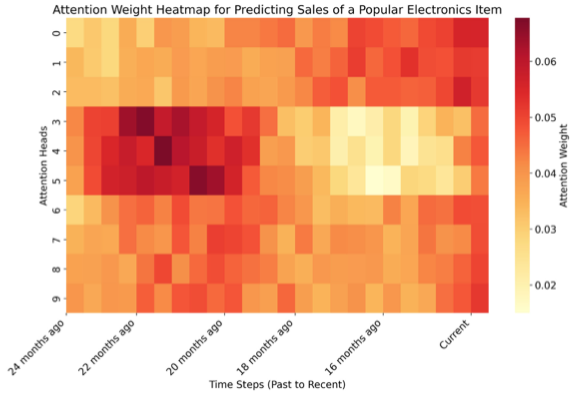

Figure 3.5: Attention weight heatmap for predicting sales of a popular electronics item. The darker the color, the higher the weight. Time in months is shown on the x-axis, and the each attention head is shown on the y-axis.[48].

be used to improve the model [20].

In the study of HA-LSTM an attention weight heatmap was used to illustrate the reasoning behind the model predictions for one popular item from the electronics category [48], which is shown in Figure (see Fig. 3.5).

The left vertical axis shows the number of attention heads and the weight of each color is described on the right. The darker the color, the more weight. The time gaps in months are shown on the horizontal axis. There are several interesting patterns that can be seen in the heatmap. First, the right side of the heatmap has a higher intensity, which means that the model is paying more attention to recent months [48]. Second, periodic patterns in attention weights correspond to annual sales cycles [48]. Third, it can be seen that different attention heads focus on different temporal patterns [48]. Heads 1-3 focus on short-term dependencies, while heads 4-6 focus on longer-term trends [48]. One reason for the robust performance of HA-LSTM in different product categories and time periods is this ability to simultaneously model various temporal scales [48].

All examples given here are just examples of what can be done to increase the interpretability of black-box models like Transformer-based models. It depends on each use case, which information and in what form is helpful to the user.

Table 3.13: Variable importance of the different variables in the Retail data are shown for the 10th, 50th, and 90th percentiles of the variable selection weights. Values larger than 0.1 are indicated by a *. The table shows, that variables like item number, store number, iten class, holidays, and past sales have a large impact for the future forecasts, as can be expected. [20]

	10%	50%	90%
(a) Static covariates			
Item num	**0.198***	**0.230***	**0.251***
Store num	**0.152***	**0.161***	**0.170***
City	0.094	0.100	0.124
State	0.049	0.060	0.065
Type	0.005	0.006	0.008
Cluster	**0.108***	**0.122***	**0.133***
Family	0.063	0.075	0.079
Class	**0.148***	**0.156***	**0.163***
Perishable	0.084	0.085	0.088
(b) Past inputs			
Transactions	0.029	0.033	0.037
Oil	0.062	0.081	0.105
On-promotion	0.072	0.075	0.078
Day of week	0.007	0.007	0.008
Day of month	0.083	0.089	0.096
Month	**0.109***	**0.122***	**0.136***
National hol	**0.131***	**0.138***	**0.145***
Regional hol	0.011	0.014	0.018
Local hol	0.056	0.068	0.072
Open	0.027	0.044	0.067
Log sales	**0.304***	**0.324***	**0.353***
(c) Future inputs			
On-promotion	**0.155***	**0.170***	**0.182***
Day of week	0.029	0.065	0.089
Day of month	**0.056***	**0.116***	**0.138***
Month	**0.111***	**0.155***	**0.204***
National hol	**0.145***	**0.220***	**0.242***
Regional hol	0.012	0.014	0.060
Local hol	**0.116***	**0.151***	**0.239***
Open	0.088	0.095	0.097

4. Summary and conclusions

In this study, I first described some challenges in the retail industry, where machine learning and, in particular, demand forecasting models could provide better solutions. The retail environment is complex, influenced by factors such as consumer behavior, multiple seasonalities, different types of promotional activities, and many other external factors. Weather and economic conditions also have an impact on the demand for products sold. Demand forecasting also serves different levels of decision making. At the strategic level, decisions such as market entry strategies, channel strategies, and location strategies need accurate long-term forecasts. At the tactical level, decisions about category management, pricing, and promotional activities need forecasts to support decision makers. At the operational level, several tasks, such as store-level inventory management and replenishment, need fast and accurate short-term forecasts.

Chapter 2 contains definitions and short model descriptions that were used later in the text. Definitions were given for time series data, different kinds of metrics used to evaluate the accuracy of the models, and some other basic formulas needed later in the text. The different methods for time series forecasting were introduced starting from the early statistical methods, like ARIMA and ETS, going through different machine learning models and deep learning models to the latest Transformer-based hybrid models and even foundation models based on them. The Transformer-based models were introduced in chronological order to give a first look at their development steps.

I started chapter 3 with some insights into the special needs and challenges for time series forecasting in retail. Then I looked briefly into different time-series forecasting methods and their performance in general. The main part was discussion about the performance of the different models in the retail environment, with a focus on the accuracy and complexity of Transformer-based models. It was pretty clear that hybrid models with Transformer-based modules perform best when it comes to accuracy. They can handle both short- and long-term dependencies on multivariate data and they can use different kinds of external variables to improve the results, both static and time varying. Also, the complexity of the Transformer-based models has decreased, but there is still some work to do on this aspect. I also discussed the interpretability

and usability of transformer-based models, as it often becomes a problem in real-life applications. Different kind of methods, like probabilistic forecasting, calculating, and showing the importance of different input variables, and several ways to highlight attention weights at different time-points, have been used to increase the interpretability of Transformer-based models.

One of the main findings was that Transformer-based models are capable of using the enormous amount of data available for the retailer to increase the performance of time series forecasts. The best models can find short- and long-term dependencies in the data, cross-correlations between different series, and they can use exogenous data to improve future forecasts. They can also use different data modalities, such as text, numbers, and pictures, in addition to raw time-series data. The Transformer-based models are also scalable for large datasets, can be used for most product categories, and have shown good results also in some special cases, like predictions for new product sales and markdown seasons. Visualizations and tables can be used to increase the interpretability of their results. The main disadvantage of Transformer-based models is the complexity, leading to long training and forecasting times. For strategic and tactical use, this might not be a problem, but for daily use, some improvements are still needed.

There are many possible models available for time-series forecasting in retail. They all have positive and negative properties, and deciding which model to use must be thought of case by case. Traditional models are often faster to train and use, tree-based models are easy to explain and accurate, and modern deep learning methods offer high accuracy with the ability to find complex inter- and intra-dependencies in the data, but with a high level of complexity. Based on the studies revealed in the previous sections, it seems pretty clear that in complicated time series forecasting tasks for retail environment Transformer-based or hybrid models are showing the best results. In addition, they are being studied and developed in a vast pace at the moment. One thing to remember is that models that perform best in the research environment are not always best in practice. In retail, the main goal is to get results that help you do better business with less cost, so sometimes a simpler model might be good enough. For simple automation tasks, where the prediction horizon is short and the need is for a simple and easy to implement solution, a traditional method or even a naive method might work. In addition, when the number of stores and products is limited, there is no need for a complicated hybrid model.

The models for time-series forecasting are developing at an enormous speed at the moment. At the same time, the retail environment gets more and more complicated, as the e-Commerce sector grows. The amount of data available to large retailers is already huge, and it is growing even faster. Together, these aspects mean that traditional

methods simply cannot handle complicated forecasting tasks in the future. Based on the arguments in this study, it seems clear that some kind of Transformer-based model will be the future in retail forecasting, most probably a foundation model trained on a larger data set.

Some interesting suggestions for future studies would be a comparison of the best models with large and complicated real-world retail data, including external variables like images and other product-related information. The lack in all the articles handled in this study is that they were made with relatively small data sets, so both accuracy and performance would be better tested. Another interesting study would be comparing all the best models in one study, instead of one modern model against several older ones, which sadly was the case in most of the papers used in this study. Also, studying the most difficult corner cases more deeply using the latest models would be interesting.

Bibliography

[1] A. A. Alioghli and F. Y. Okay. Enhancing multivariate time-series anomaly detection with positional encoding mechanisms in transformers. *The Journal of Supercomputing*, 81(282), 2024.

[2] T. Antamis, A. Drosou, T. Vafeiadis, A. Nizamis, D. Ioannidis, and D. Tzovaras. Interpretability of deep neural networks: A review of methods, classification and hardware. *Neurocomputing*, 601, 2024.

[3] V. Bhogade and B. Nithya. Time series forecasting using transformer neural network. *International journal of computers and applications*, 46(10):880–888, 2024.

[4] E. Bilgic, O. Cakir, M. Kantardzic, Y. Duan, and G. Cao. Retail analytics: store segmentation using rule-based purchasing behavior analysis. *The International Review of Retail, Distribution and Consumer Research*, 31(4):457–480, 2021.

[5] C. Brackmann, M. Hutsch, and T. Wulfert. Identifying application areas for machine learning in the retail sector. *SN Computer science*, 4(426), 2023.

[6] D. Brykin. Sales forecasting models: Comparison between ARIMA, LSTM and Prophet. *Journal of Computer Science*, 10(20):1222–1230, 2024.

[7] R. Caetano, J. M. Oliveira, and P. Ramos. Transformer-based models for probabilistic time series forecasting with explanatory variables. *Mathematics*, 13(814), 2025.

[8] D. Cao and S. Zhang. AD-Autoformer: decomposition transformers with attention distilling for long sequence time-series forecasting. *The Journal of Supercomputing*, 80:21128–21148, 2024.

[9] Y. Chen, S. Liu, J. Yang, H. Jing, W. Zhao, and G. Yang. A joint time-frequency domain transformer for multivariate time series forecasting. *Neural Networks*, 176, 2024.

[10] Y. Chen, X. Xie, Z. Pei, W. Yi, C. Wang, W. Zhang, and Z. Ji. Development of a time series e-Commerce sales prediction method for short-shelf-life products using GRU-LightGBM. *Applied Sciences*, 14(866), 2024.

[11] C. Feng, L. Huan, and D. Krompass. General time transformer: an encoder-only foundation model for zero-shot multivariate time series forecasting. *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, 1, 2024.

[12] R. Fildes, S. Ma, and S. Kolassa. Retail forecasting: Research and practice. *International Journal of Forecasting*, 38:1283–1318, 2022.

[13] E. C. Griffin, B. B. Keskin, and A. W. Allaway. Clustering retail stores for inventory transshipment. *European Journal of Operational Research*, 311:690–707, 2023.

[14] N. Gulcan and D. T. Aksu. Demand forecasting in clearance season for fast fashion retail: a residual learning approach. *International Journal of Clothing Science*, 37(2):308–321, 2025.

[15] Y. Han, S. R. Chandukala, and S. Li. Impact of different types of in-store displays on consumer purchase behavior. *Journal of Retailing*, 98:432–452, 2022.

[16] S. Lee, J. Hong, L. Liu, and W. Choi. TS-Fastformer: Fast transformer for time-series forecasting. *ACM Transactions of Intelligent Systems and Technology*, 15(2), 2024.

[17] D. Li, Q. Liu, D. Feng, and Z. Chen. A medium- and long-term residential load forecasting method based on discrete cosine transform-FEDformer. *Energies*, 17(3676), 2024.

[18] J. Li, B. Lin, P. Wang, Y. Chen, X. Zeng, X. Liu, and R. Chen. A hierarchical RF-XGBoost model for short-cycle agricultural product sales forecasting. *Foods*, 13(2936), 2024.

[19] Y. Liang, H. Wen, Y. Nie, Y. Jiang, M. Jin, D. Song, S. Pan, and Q. Wen. Foundation models for time series analysis: A tutorial and survey. *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining August 25-29 2024 Barcelona Spain*, 1, 2024.

[20] B. Lim, S. Ã. Arik, N. Loeff, and T. Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37:1748–1764, 2021.

[21] J. Liu, Z. Pang, and L. Qi. Dynamic pricing and inventory management with demand learning: A Bayesian approach. *Computers and Operations Research*, 124, 2020.

[22] L. Liu, X. Wang, X. Dong, K. Chen, Q. Chen, and B. Li. Interpretable feature-temporal transformer for short-term wind power forecasting with multivariate time series. *Applied Energy*, 374, 2024.

[23] S. Liu, H. Yu, C. Liao, J. Li, W. Lin, A. X. Liu, and S. Dustbar. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. *Proceedings of the International conference on learning representations 25-29 April 2022*, 1, 2022.

[24] X. Liu and W. Wang. Deep time series forecasting models: A comprehensive survey. *Mathematics*, 12(1504), 2024.

[25] Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, and M. Long. iTransformer: Inverted transformers are effective for time series forecasting. *Conference paper at ICLR 2024*, 1, 2024.

[26] S. Mejia and J. Aguilar. A demand forecasting system of product categories defined by their time series using a hybrid approach of ensemble learning with feature engineering. *Computing*, 106:3945–3965, 2024.

[27] J. M. Oliveira and P. Ramos. Evaluating the effectiveness of time series transformers for demand forecasting in retail. *Mathematics*, 12(2728), 2024.

[28] O. Pak, M. Ferguson, O. Perdikaki, and S.-M. Wu. Optimizing stock-keeping unit selection for promotional display space at grocery retailers. *Journal of Operational Management*, 66:501–533, 2020.

[29] M. Qi, H.-Y. Mak, and Z.-J. M. Shen. Data-driven research in retail operations - a review. *Naval Research Logistics*, 67:595–616, 2020.

[30] A. Rafi, G. N. Rodrigues, N. H. Mir, S. M. B. M. F. Mridha, R. Islam, and Y. Watanobe. A hybrid temporal convolutional network and transformer model for accurate and scalable sales forecasting. *IEEE Open Journal of the Computer Society*, 6:380–391, 2025.

[31] M. Rajendran and B. Hong. Autoregressive multimodal transformer for zero-shot sales forecasting of fashion products with exogenous data. *Applied Intelligence*, 55(108), 2024.

[32] P. Ramos and J. M. Oliveira. Robust sales forecasting using deep learning with static and dynamic covariates. *Applied System Innovation*, 6(85), 2023.

[33] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36:1181–1191, 2020.

[34] M. Sousa, A. Loureiro, and V. Migueis. Predicting demand for new products in fashion retailing using censored data. *Expert systems with applications*, 259, 2025.

[35] K. Swaminathan and R. Venkitasubramony. Demand forecasting for fashion products: A systematic review. *International Journal of Forecasting*, 40:247–267, 2024.

[36] M. Teixeira, J. M. Oliveira, and P. Ramos. Enhancing hierarchical sales forecasting with promotional data: A comparative study using ARIMA and deep neural networks. *Machine learning and knowledge extraction*, 6:2659–2687, 2024.

[37] G. Tong, Z. Ge, and D. Peng. RSMformer: an efficient multiscale transformer-based framework for long sequence time-series forecasting. *Applied Intelligence*, 54:1275–1296, 2024.

[38] H. Tong, L. Kong, J. Liu, S. Gao, Y. Xu, and Y. Chen. Segmented frequency-domain correlation prediction model for long-term time series forecasting using transformer. *IET Software*, 2024, 2024.

[39] J. Wang, W. K. Chong, J. Lin, and C. P. T. Hedenstierna. Retail demand forecasting using spatial-temporal gradient boosting methods. *Journal of Computer Information Systems*, 2023.

[40] F. Weber and R. Schutte. A domain-oriented analysis of the impact of machine learning-the case of retailing. *Big data and cognitive computing*, 3(11), 2019.

[41] A. P. Wellens, R. N. Boute, and M. Udenio. Simplifying tree-based methods for retail sales forecasting with explanatory variables. *European Journal of Operational Research*, 314:523–539, 2024.

[42] C. Yang, Y. Wang, B. Yang, and J. Chen. GRAformer: A gated residual attention transformer for multivariate time series forecasting. *Neurocomputing*, 581, 2024.

[43] X. Yao, T. Sanl, and A. Gulcu. Markdown optimization for multiple products with a common markdown budget constraint. *Journal of Revenue and Pricing Management*, 14(6):442–450, 2015.

[44] H. Ye, Y. Zheng, X. Li, L. Dong, W. Huang, J. Wang, S. Yan, H. Lou, P. Yan, S. Zhang, X. Li, Y. Ling, X. Huang, and Y. Pan. A transformer-based forecasting model for $f_1$0.7 index and its application study on the Chinese Langfang dataset. *Advances in Space Research*, 74:6309–6324, 2024.

[45] C. Ying and J. Lu. TFEformer: Temporal feature enhanced transformer for multivariate time series forecasting. *IEEE Access*, 12:153694–153708, 2024.

[46] A. Zeng, M. Chen, L. Zhang, and Q. Xu. Are transformers effective for time series forecasting? *AAAI Conference on Artificial Intelligence*, 37:11121–11128, 2023.

[47] Z. Zeng, W. Yang, M. Wang, M. Zhu, and T. Feng. Chain drugstore sales prediction method based on fusion of self-attention mechanism and LightGBM. *J syst sci syst eng*, 0(0):1–18, 2025.

[48] X. Zhang, P. Li, X. Han, Y. Yang, and Y. Cui. Enhancing time series product demand forecasting with hybrid attention-based deep learning models. *IEEE Access*, 12:190079–190091, 2024.

[49] H. Zhou, K. Chen, and S. Wang. Two-period pricing and inventory decisions of perishable products with partial lost sales. *European Journal of Operational Research*, 310:611–626, 2023.

[50] S. Zhu, J. Zheng, and Q. Ma. MR-Transformer: Multiresolution transformer for multivariate time series prediction. *IEEE transactions on neural networks and learning systems*, 36(1):1171–1183, 2025.