Deep Learning based Forecasting: a case study from the online fashion industry

Manuel Kunz
Zalando SE
manuel.kunz@zalando.de

Stefan Birr
Zalando SE
stefan.birr@zalando.de

Mones Raslan
Zalando SE
mones.raslan@zaland.de

Lei Ma
Zalando SE
lei.ma@zalando.de

Zhen Li
Zalando SE
peter.zhen@zalando.de

Adele Gouttes
Zalando SE
adele.gouttes@zalando.de

Mateusz Koren
Zalando SE
mateusz.koren@zalando.de

Tofigh Naghibi
Zalando SE
tofigh.naghibi@zalando.de

Johannes Stephan
Zalando SE
johannes.stephan@zalando.de

Mariia Bulycheva
Zalando SE
mariia.bulycheva@zalando.de

Matthias Grzeschik
Zalando SE
matthias@zalando.de

Armin Kekić *
Max Planck Institute for Intelligent Systems
Tübingen, Germany
armin.kekic@mailbox.org

Michael Narodovitch
Zalando SE
michael.narodovitch@zalando.de

Kashif Rasul *
Morgan Stanley
kashif.rasul@gmail.com

Julian Sieber
Zalando SE
julian.sieber@zalando.ie

Tim Januschowski
Zalando SE
tim.januschowski@zalando.de

Abstract

Demand forecasting in the online fashion industry is particularly amendable to global, data-driven forecasting models because of the industry's set of particular challenges. These include the volume of data, the irregularity, the high amount of turn-over in the catalog and the fixed inventory assumption. While standard deep learning forecasting approaches cater for many of these, the fixed inventory assumption requires a special treatment via controlling the relationship between price and demand closely. In this case study, we describe the data and our modelling approach for this forecasting problem in detail and present empirical results that highlight the effectiveness of our approach.

1 Introduction

Forecasting problems in the fashion industry and in particular, in its online variant, come with a particular set of challenges and down-stream use-cases which we illustrate in our article. The arguably most peculiar challenge in the fashion industry stems from a fixed inventory assumption where a meaningful part of the inventory arrives at the beginning of a season (having been ordered much ahead of the season) and it cannot be re-ordered throughtout the season. Hence, for this non-reorderable part of the assortment, forecasting is not primarily used for replenishment

*Work done while at Zalando

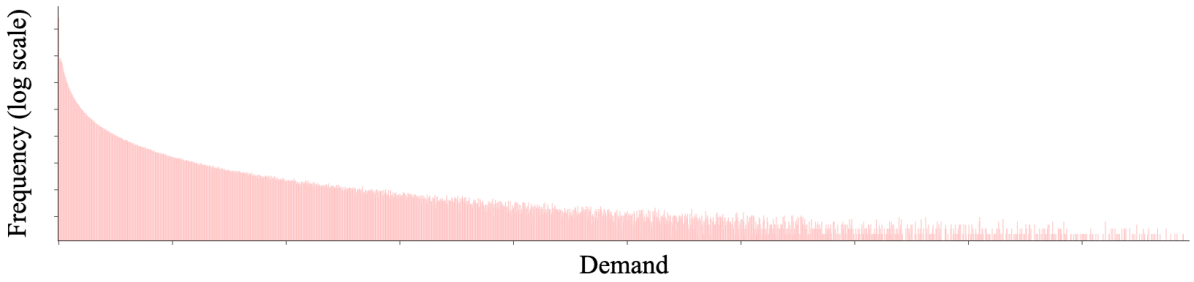


Figure 1: Histogram of demand (in log scale) per article: The histogram shows the heavy tailed demand distribution. On the x-axis is demand starting from 0, on the y-axis the frequency of the demand occurrence over one time unit.

use-cases as is the case in say, grocery (e.g., [14]). Instead, a primary use-case for forecasting is pricing/discounting (e.g., [26]). Forecasting provides the input for a down-stream mixed integer optimization problem that sets the prices with a notion of optimality in a classical forecast-then-optimize setting. In the online fashion industry, solving the pricing problem is a means to handle inventory risk management because pricing presents a major lever to reduce the risk to be overstocked at the end of the season. Complications that arise from the fixed inventory assumptions include the prominence of cold start problems (a large part of the catalogue is renewed each season) and short history. Other complications are commonalities also present in other online industries, namely a large catalogue with millions of stock keeping units (SKUs) which on the one hand results in a large overall scale of the forecasting problem and on the other hand in sparsity on the SKU level. We present examples and illustrations of the complications in a detailed (necessarily anecdotal) description of the data and the implied challenges for forecasting.

The main contribution of our article is the description of the approach that Zalando SE, a market-leading fashion retailer in Europe, has taken to address the forecasting problems that arise in pricing. Demand forecast subject to differing pricing levels poses the primary forecasting challenge that we aim to address. Given the complications described above, a natural choice is a flexible and hence, highly-parameterized, global [19] forecasting model that allows to learn complex patterns across products and seasons in a data-driven way. We describe a formerly unpublished deep learning based forecasting model relying on the transformer [50] architecture. It has been in use at Zalando in different versions since 2019 and hence is, to the best of our knowledge, among the first examples of attention-based forecasting models in production in industry.

Our article is structured as follows. We start by describing the data and associated covariates in Section 2. In Section 3, we describe our model in detail and explain how we address the problems that arise in our context. We place a particular focus on the importance of the rich covariate structure available in our forecasting problems. The importance of price as a covariate is fundamental and we describe how we incorporate inductive biases to learn appropriate price elasticities into our model. In Section 4, we include empirical results and compare our model performance against other approaches. We also try to explain why a transformer-based architecture performs better for our use-case, by testing for scaling laws. We discuss related work in Section 5, where we mention advantages and limitations of our approach, contrasting our approach with publicly available methods. In Section 6, we comment on practical challenges such as monitoring forecasting accuracy in production, its down-stream usage and deciding how to re-train. We conclude in Section 7, where we also discuss avenues for future work that hopefully inspire future research.

2 Data for forecasting at Zalando: An Overview

Zalando is active in 25 European markets, its catalog comprises of >6500 brands with >50M active customers. This amounts to 1.6 million articles on the Zalando platform (as of Q3 2021). We are interested primarily in modelling the future demand of an article, where the demand is the quantity of items customers would buy in one time unit. Demand at Zalando follows a typical long-tail distribution, see Figure 1. Note that we cropped the histogram on the right and cannot provide units for the x-axis and y-axis for confidentiality reasons. Most articles do not sell on most days, but there are a few high-selling products. Similar demand characteristics are available in other real-world examples [42, 29, 25].

Complementing the aggregate statistics, Figure 2 gives examples for what individual time series look like. In Figure 3, we show additionally the stock levels in red dashed lines for an example of an article that has a complicated stock

Feature	Dim	Category	Value Type	Transformation	available in future?
Brand	10	static-global	categorical	embedded	y
Commodity Group	10	static-global	categorical	embedded	y
Discount	14	dynamic-int'l	numeric	–	n
Black Price	14	static-int'l	numeric	logarithm	y
Sales	14	dynamic-int'l	numeric	logarithm	n
Stock	1	dynamic-global	numeric	logarithm	n
Stock Uplift	1	dynamic-global	numeric	logarithm	n

Table 1: Examples for covariates available for the forecasting problem at Zalando.

availability. These examples illustrate the difficulties in extracting patterns from the observational time horizon of a single article.

2.1 Sales to Demand Translation

Note that, similar to other demand forecasting problems (e.g., [14, 8, 51, 13]), we do not observe demand, but only sales. Demand corresponds to what the customers want to purchase, and materialises only when enough stock is available. In our case, we have a stock signal available which indicates when sales are strictly less than demand because no stock is available. While it is possible to handle such cases by marking the demand data as *missing* and let the forecast model handle this case (see [6] for an example), here we adopt a different approach by pre-processing our sales data and impute demand.

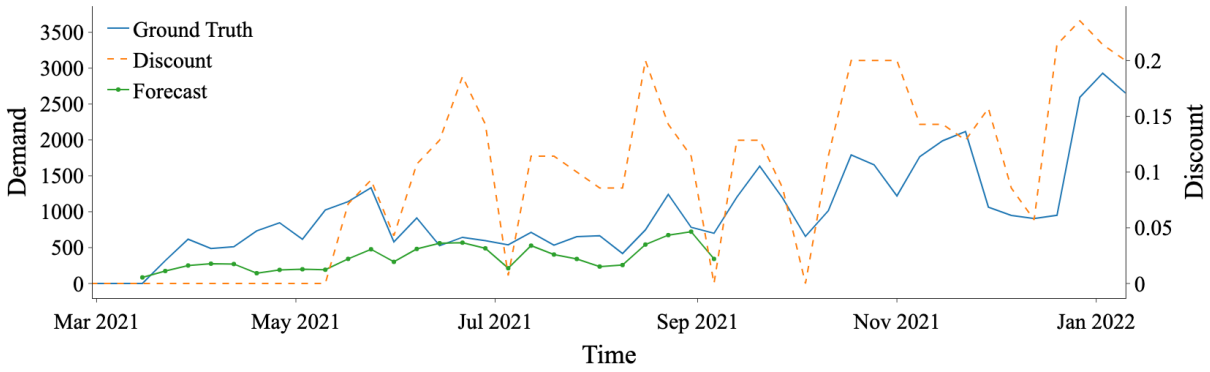
We can take advantage on the fact that in fashion, articles typically come in different sizes. Note that we do not desire to price differently different sizes, and so the forecasting task requires to predict demand of a single fashion item aggregated over all sizes. If all sizes of an article are available or *in stock* over one time unit, the demand of that article equals the materialized/observed sales. If some sizes are out of stock, the demand is in fact (partially) unobserved and we cannot directly obtain it from the sales data. To remediate this, we first define the demand q of an article i with k sizes as a random variable $\mathbf{q_i} = \{q_{i1}, q_{i2}, \ldots, q_{ik}\}$. In order to infer the expected demand over all sizes $n_i = \sum_{n=1}^{k} q_{in}$, we assume $\mathbf{q_i}$ to be multinomially distributed with $\mathbf{q_i} \sim \text{Multinomial}(n_i, \mathbf{p_i})$ and $\mathbf{p_i} = \{p_1, p_2, \ldots, p_k\}$ the discrete probability distribution over the k sizes of article i. If we further assume to know $\mathbf{p_i}$, we can compute n_i based on a partial observation $\{q_{i1}, q_{i2}, \ldots, X_{ij}, \ldots, q_{ik}\}$, with X_{ij} representing the missing demand observations. This procedure only requires to learn $\mathbf{p_i}$ from observational data in intervals with no stock-outs, assuming $\mathbf{p_i}$ to be time invariant. We then use the learned $\mathbf{p_i}$ to translate sales to demand, and we use this imputed demand in model training. Note that, when adopting such a preprocessing step, special care has to be taken by not evaluating our model on extrapolated data, but rather exclude this data from evaluation. In Figure 4, we depict the process described above and Figure 5 visualizes the result of our model applied to a specific article over time.

2.2 Description of Covariates

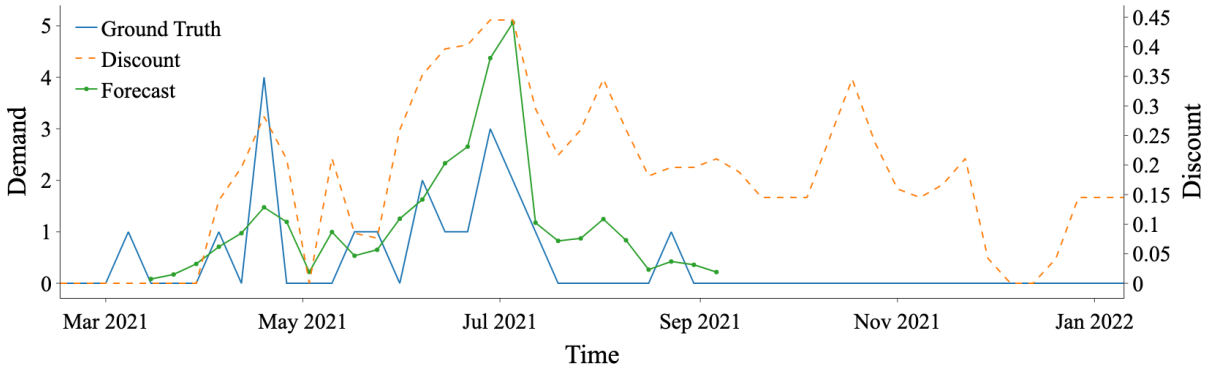
The covariates that we use in our forecasting model can be grouped into the following categories: *static-global*: time-independent & market-independent; *dynamic-global*: time-dependent & market-independent; *dynamic-international*: time-dependent & market-specific; and *static-international*: time-independent & market-specific. We give a non-exhaustive list for the covariates in use, described in Table 1 in more detail for illustration purposes.

The *static-global* covariates Brand and Commodity Group categorize an article. The later provides information about the part of the assortment (e.g, Men - Sportswear). They do not change over time and across countries. The Black Price is the recommended retail price and the initial price an article receives. It is the reference price for every discount applied. We use its logarithm to prevent scaling issues typical in global forecasting models [19]. Black Price and Discount are *international* covariates, containing potentially different values in different countries. The Black Price of an article does not change over time, but Discount might change which makes it *dynamic*.

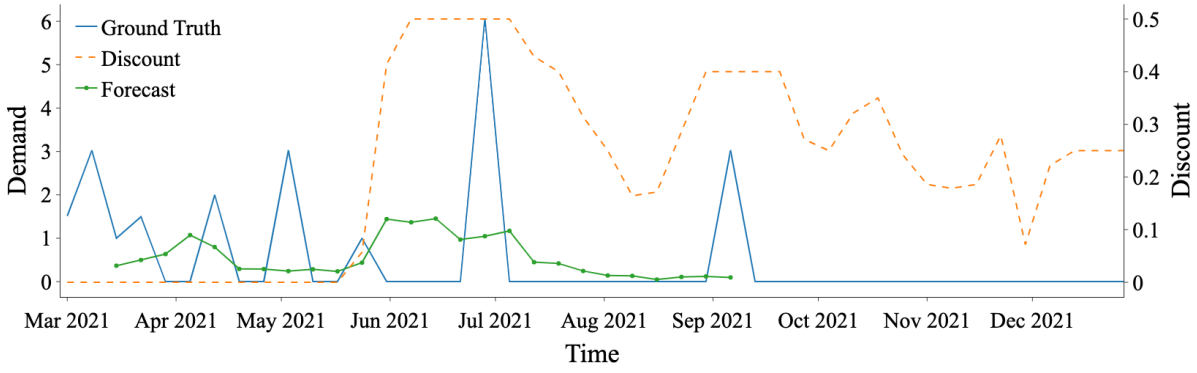
Note that not all covariates are available for the forecast horizon. For example, Stock or Discount are only available historically. We include them because they either help to "explain away" certain historical effects or because we set them explicitly in the forecast horizon to obtain what-if forecasts. Other covariates, like the Stock Uplift, are only available as an estimate for the future time horizon. The Stock Uplift expresses for each time unit the increase in stock of an article, caused by supplier deliveries or customer returning their pruchase. This value is available for the past but only partially available for the future and needs to be estimated based on supplier delivery data. Both Stock and Stock Uplift are *global*, because stock is shared across countries.



(a) Example of a cold start problem.



(b) Example of short history.



(c) Example of a sparsity/intermittency.

Figure 2: Examples for complications occurring in practice: The dashed orange line gives discount level (the higher, the more discount), blue is observed (ground truth) demand and green with markers is the forecast. Note that everything left to the first green dot is the available demand history. We have hence articles with little to no history, see Figures 2a and 2b as well as sparse time series, Figure 2c.

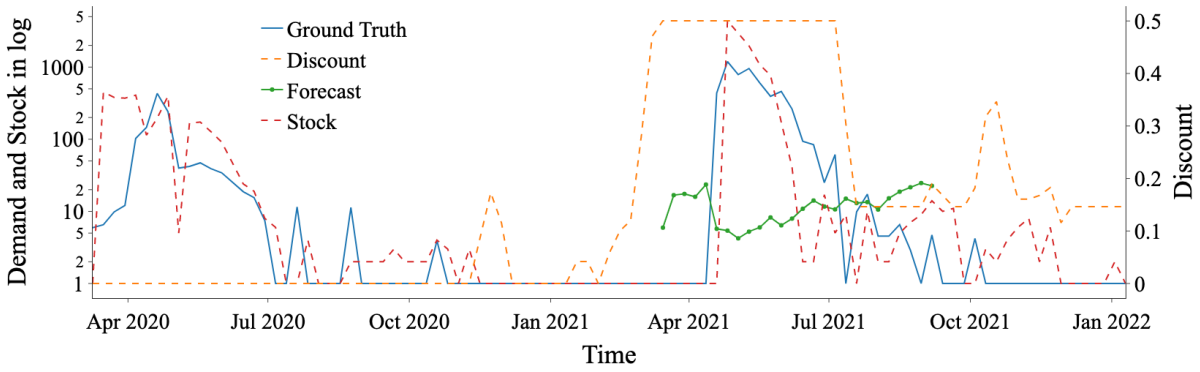


Figure 3: Example for a challenging stock situation: The red dashed line represents stock levels over time. The article was out of stock for a longer time period until it was replenished in April 2021. It can be seen that the ground truth is highly correlated with stock availability of an article which provides additional modelling challenges as discussed in Section 2.1

3 Demand Forecast Model

The demand forecasting model used within Zalando Pricing is a global [33, 19] forecasting model. This means that a single forecasting models is trained over the entire Zalando article[2] assortment and used to provide article specific predictions on a weekly time resolution for a horizon of 26 weeks. Additionally, our forecast is used to decide on the discounts applied to each article: we need to forecast future demand for multiple scenarios. Before we describe the forecasting approach in detail, we first formalize the forecasting problem.

3.1 Problem Formalization

For any timeseries x, $x_{0:T}$ is short-hand for $[x_0, x_1, \ldots, x_T]$. The observational timeseries data of an article i at time t starting at 0 is given by $\{q_{i0:t}, d_{i0:t}, z_{i0:t}\}$, where q denotes the demand. d corresponds to the discount, which is the percentage of price reduction relative to the article's recommended retailer price; and z a set of article specific covariates. Our object of interest is

$$P(q_{i,t+1:t+h}|q_{i,0:t}, d_{i,0:t+h}, z_{i,0:t+h}; \theta) \,, \tag{1}$$

that is the probability distribution of demand in the forecast horizon $t+1 : t+h$ conditioned on (among others) discounts in the forecast horizon. We are interested in identifying appropriate θ. Note that (1) is a simplification. More generally, we should be interested on the one hand in a multi-variate version so that we model cross-article effects and, on the other hand, we should actually be interested in

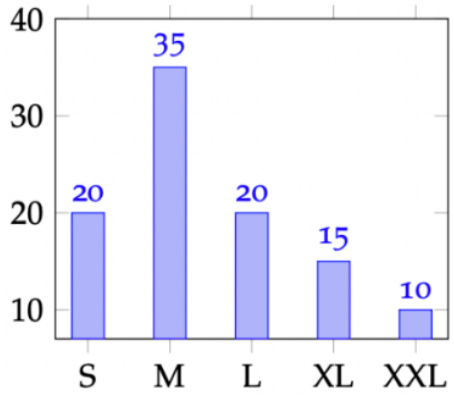
$$P(q_{i,t+1:t+h}|\mathrm{do}(d_{t+1:t+h}), q_{0:t}, d_{0:t}, z_{0:t+h}; \theta) \,, \tag{2}$$

where the do operator denotes the intervention of setting a price/discount to a certain level (see e.g., [37] for the notation and additional background). This is because we are interested in taking down-stream decisions on prices so the (causal) relationship between price/discount and demand is of fundamental importance. We leave a more comprehensive causal treatment for future work and mention that in practice, we approximate (1) by a point forecast and point to [16] for a discussion of a probabilistic extension of the model described here. However, we remark that (1) is a simplification of our actual model in the sense that we provide a multi-variate forecast because we forecast all markets in which Zalando is active in simultaneously. So our demand time series actually has an additional index per market. However, for ease of exposition, we do not further elaborate on this complication.

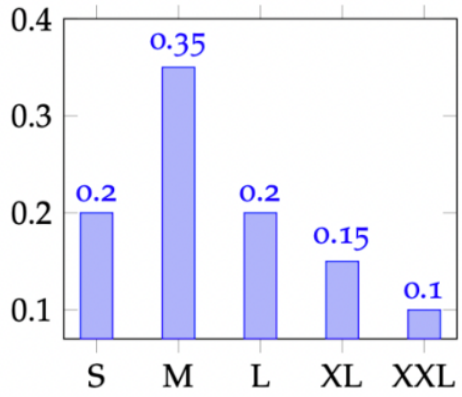
3.2 Model Architecture

For the demand forecast model we rely on an encode/decoder [46] approach. This is primarily motivated by the fact that we have some covariates that are not available in the forecast horizon and others that are available only for the future, see Table 1. The encoder/decoder architecture is particularly amendable for this setting, see e.g., [13, 51].

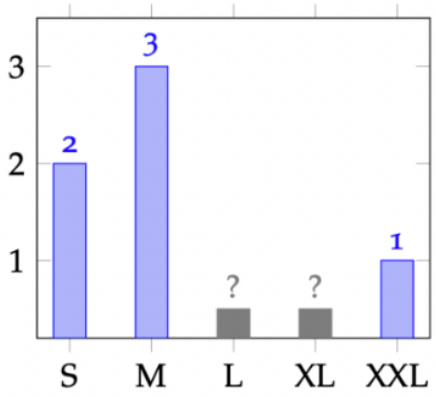
[2]A sellable article offered in the Zalando fashion store.



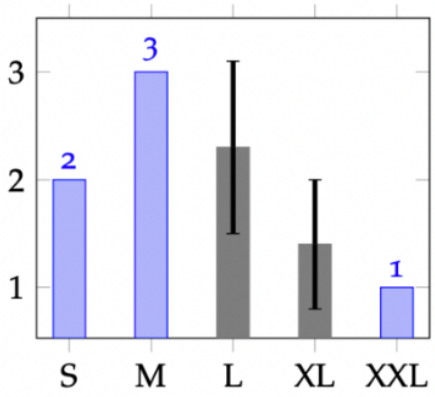
(a) Total observed sales over all weeks with full size availability.



(b) Probability distribution over sizes ($\vec{p}$ parameter of Multinomial).



(c) Sales for week with sizes L and XL missing.



(d) Estimates for sales in sizes L and XL.

Figure 4: Sales to demand translation for an article with 5 sizes: In (a), historical sales observations of an article over weeks with all sizes available are aggregated. Based on (a), we obtain in (b) the articles's empirical probability distribution over sizes. In (c), the weekly demand of an article with missing sizes L and XL is illustrated. The unobserved demand for L and XL is inferred in (d), given the observed demand of the other sizes for that week and the empirical distribution computed in (b).

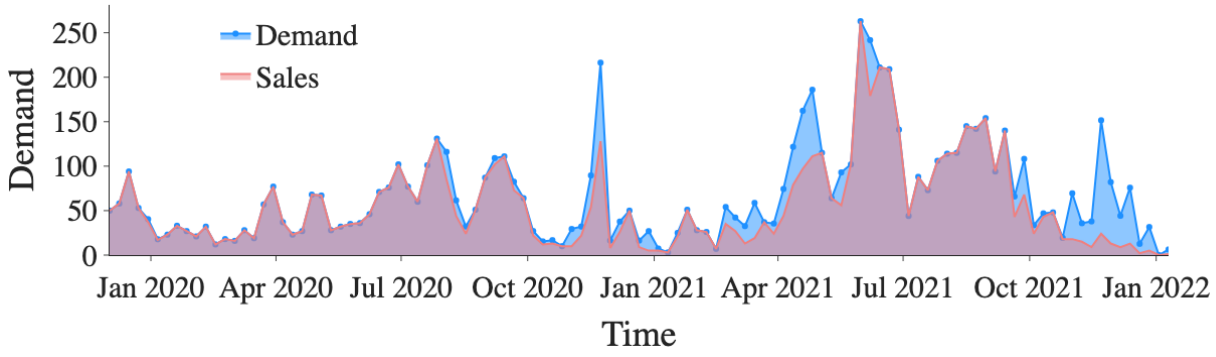


Figure 5: Demand vs sales for a single article. In weeks with full availability, sales and demand overlaps. If a size or the full article becomes unavailable, demand becomes an estimate larger than the observed sales.

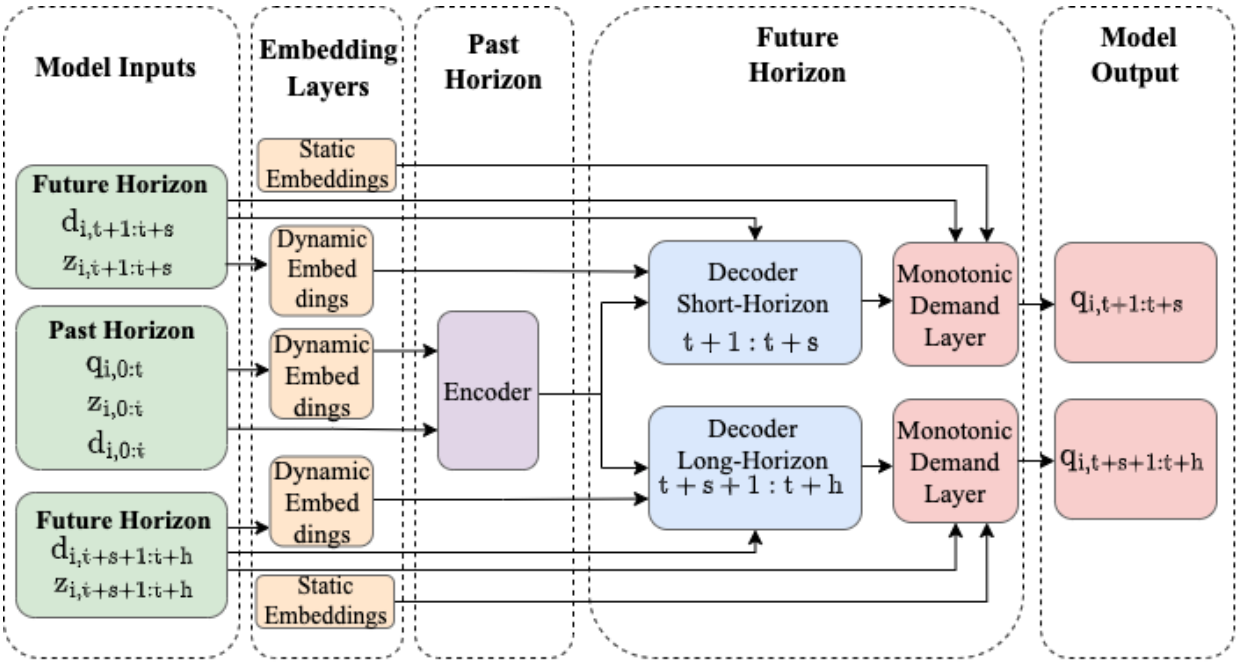


Figure 6: Demand Forecaster Architecture: Encoder handles the observational time series, decoder incorporates the future covariates. Future discounts are directly provided to the output (Monotonic Demand) layer, skipping decoder and encoder.

For the model architecture itself, we base it on the standard Transformer architecture and we depict it in Figure 6. We have additional embedding [32] layers for some of the covariates. Our main modelling innovations are two-fold. First, we specialize our decoder into a short- and a long-term horizon. This is because our down-stream application requires particular focus on the first forecast weeks for which the pricing decisions are taken, whereas the further out future only plays a secondary importance. We further adapt the training scheme to account for the focus on the first weeks. We discuss this in Section 3.4. Second, we need to control the relationship between price and demand closely given the primary usage of the forecasts in pricing. We achieve this by enforcing a monotonic relationship between price and demand which we parameterize flexibly as a piece-wise linear function (Section 3.2.6).

3.2.1 Input preparation

In the forward pass, the model receives the data of a single article shown as Model Inputs in Figure 6. All features are stacked into a single input tensor of dimension $\mathbb{R}^{t \times u \times v}$, where t is the time dimension, u the batch size and v the dimension of the covariate vector.

The high dimensional but sparse categorical covariates are passed through the embedding layers before being forwarded to the encoder, decoder and monotonic demand layer. We distinguish between two types of embeddings, dynamic and static. Dynamic embeddings are time dependent and provided as input to the encoder and decoder. Static embeddings are time independent and provided as additional input to the monotonic demand layer. The temporal structure of the time series is represented by positional encoding, introduced in Section 3.2.3.

The decoder receives all covariates associated with the forecasting time horizon, the associated embeddings and the output of the encoder. Encoder and decoder consist of various multihead attention layers [3, 50], to represent temporal correlations across the covariates. Instead of feeding future discounts into the decoder, we directly provide them to the monotonic demand layer, see Section 3.2.6, which allows us to predict demand as a monotonically increasing function of discount.

3.2.2 Encoder

The task of the encoder is to learn a representation of the observational article data, optimized for predicting future demand. The encoder model architecture is based on multi-head attention [50]. In the following, we recall the concepts and explain how to translate them to the timeseries setting. In a first step, we project the input data $\{d_{0:t}, q_{0:t}, z_{0:t}\}$ into three (Q, K, V) [3] tensors. Therefore, for each article i, the data is transformed into the matrix $\eta_{0:t}$:

$$\eta_{i,0:t} = \begin{bmatrix} q_{00} & q_{01} & \cdots & q_{0t} \\ d_{10} & d_{11} & \cdots & d_{1t} \\ z_{20} & z_{21} & \cdots & z_{2t} \\ \vdots & \vdots & \ddots & \vdots \\ z_{v0} & z_{v1} & \cdots & z_{vt} \end{bmatrix}, Q = \phi_q(\eta_{i,0:t}), K = \phi_k(\eta_{i,0:t}), V = \phi_v(\eta_{i,0:t}) \tag{3}$$

Each column in $\eta_{i,0:t}$ represents a covariate vector with demand, discount and additional covariates of an article i at time $T \in [0, \ldots, t]$. The matrix $\eta_{i,0:t}$ is then multiplied by three learned weight matrices, implemented as linear layers ϕ_q, ϕ_k, ϕ_v. Scaled dot product attention (4) and multi-head attention (5) is then applied to Q, K and V, following [50].

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{4}$$

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O \\ \text{head}_j &= \text{Attention}(QW_j^Q, KW_j^K, VW_j^V) \end{aligned} \tag{5}$$

In the original transformer architecture, d_{model} represents the embedding dimension of an embedded word vector. Applied to the time series data set in 3, d_{model} is defined by the number of rows in $\eta_{i,0:t}$, or expressed differently, by the covariate vector of dimension v, so $v = d_{model}$. Similar as in the original transformer architecture, the number of attention heads h, the dimension of the linear layers d_k and the dimension of the embedding layers were chosen to match $d_{model} = h \cdot d_k$. The linear layers ϕ_q, ϕ_k, ϕ_v were specified with same input and output dimension to produce Q, K and V tensors of shape $h \times (t + 1) \times d_k$. The remaining operations follow (4),(5). Note that the computation of the h attention heads allows parallelization and is most efficiently implemented by applying vectorized tensor operations [36]. The encoder contains multiple stacked multi-head attention layers with two sublayers and residual connections [18] as specified in [50]. Dropout [43] and Layer Normalization [2] is applied on the output of each sublayer.

3.2.3 Positional Encoding

Positional encoding is a fundamental part of the original transformer architecture. For each position in the input embeddings, a positional encoding of dimension d_{model} is computed and summed up with the corresponding word embedding at this position [50]. This approach is not transferable to the time series data introduced in Section 2 because

[3] Q = Query, K = Key, V = Value as in the standard transformer nomenclature; note that this clashes with the notation that we introduced before unfortunately, but it is standard in the transformer literature.

of the heterogeneous covariate structure. Instead, a positional encoding feature with embedding dimension e_{dim} is computed and integrated as an independent covariate to $\eta_{i,0:t}$.

$$\begin{aligned} \text{pos}_{0:t} &= \begin{bmatrix} \text{pos}_0^T & \text{pos}_1^T & \cdots & \text{pos}_n^T & \cdots & \text{pos}_t^T \end{bmatrix} \\ \text{pos}_n &= [\sin(p_{n,0}), \cos(p_{n,1}), \ldots, \sin(p_{n,e_{\text{dim}}-2}), \cos(p_{n,e_{\text{dim}}-1})] \\ p_{n,m} &= 2 \cdot \pi \cdot f_m \cdot n \\ f_m &= \frac{2 \cdot m + 1}{e_{\text{dim}} \cdot t_{\text{dim}}} \end{aligned} \tag{6}$$

The positional encoding feature is defined by $pos_{0:t}$ in (6). Each position in $\text{pos}_{0:t}$ is a sequence of sine and cosine values computed at different frequencies f_m for each embedding dimension in the positional encoding. Each frequency is defined relative to $t_{dim} = 52$, representing annual cycles given weekly time resolution.

3.2.4 Padding and Masking

The original transformer architecture applies zero padding and masking to sequences with variable length in order to reach the defined input embedding dimension and to avoid a look-ahead in the decoder. In time series forecasting, only the time dynamic covariates need to be zero padded and masked which becomes especially important if slicing is applied in order generate multiple training samples out of one observational time series. Masking is then applied to the result of the dot product of the Q and V tensors in order to prevent the padded dimensions influencing the attention scores. In the case of the demand forecasting model, not only zero padded values are masked out but also all data points with zero stock availability. This is motivated by issues inferring demand on article data with zero stock availability.

3.2.5 Decoder

The task of the decoder is to compute for each time unit in the forecast horizon a future state. The matrix $\eta_{i,t+1:t+h}$ 7 represents the input to the decoder. Each column vector $\eta_{i,T}$ in $\eta_{i,t+1:t+h}$ contains the known future covariates of an article i in future week $T \in [t+1, \ldots, t+h]$ and the last observed discount d_t and demand q_t at time t. In order to compute the future state at time T, the decoder receives the encoder's output $\gamma_{i,0:t}$ and the future covariate vector $\eta_{i,T}$ (8) as input. Similar to the encoder, the decoder contain multi-head attention layers but with a specific attention mechanism (9) applied.

$$\gamma_{i,0:t} = \begin{bmatrix} e_{00} & e_{01} & \cdots & e_{0t} \\ e_{10} & e_{11} & \cdots & e_{1t} \\ e_{20} & e_{21} & \cdots & e_{2t} \\ \vdots & \vdots & \ddots & \vdots \\ e_{k0} & e_{k1} & \cdots & e_{kt} \end{bmatrix}, \eta_{i,t+1:t+h} = \begin{bmatrix} d_{0t} & d_{0t} & \cdots & d_{0t} \\ q_{1t} & q_{1t} & \cdots & q_{1t} \\ z_{2,t+1} & z_{2,t+2} & \cdots & z_{2,t+h} \\ \vdots & \vdots & \ddots & \vdots \\ z_{k,t+1} & z_{k,t+2} & \cdots & z_{k,t+h} \end{bmatrix} \tag{7}$$

$$Q = \phi_q(\eta_{i,T}), K = \phi_k(\gamma_{i,0:t}), V = \phi_v(\gamma_{i,0:t}) \tag{8}$$

In (8), the decoder input is passed through a linear layer ϕ, similar as in (3). Note that the dimension of $\eta_{i,T}$ and $\gamma_{i,0:t}$ are different, causing Q to be different in dimension compared to K and V. This is solved by repeating values of Q to align in dimension, before stacked together and passed through linear layer τ computing the inner product across past and future.

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{\tau(Q, K)}{\sqrt{d_k}})V \tag{9}$$

The slightly changed attention formula shown in (9) is applied for each column vector in $\eta_{i,t+1:t+h}$, computing the future states in the forecast horizon $t+1 : t+h$. Different to the original transformer architecture, the decoder is non-autoregressive, allowing to compute each future state in parallel. This architectural choice was necessary to enforce independence[4] between forecast weeks. Similar as in the encoder, normalization, dropout and residual connections are applied. The decoder output $\kappa_{i,t+1:t:h}$ is passed with the encoder state $\gamma_{i,0:t}$ and the future discounts $d_{i,t+1:t+h}$ to the Monotonic Demand Layer in order to predict the future demand $q_{i,t+1:t+h}$.

[4]The independence assumption between future weeks was made in order simplify the downstream pricing models, consuming the output of the demand forecaster.

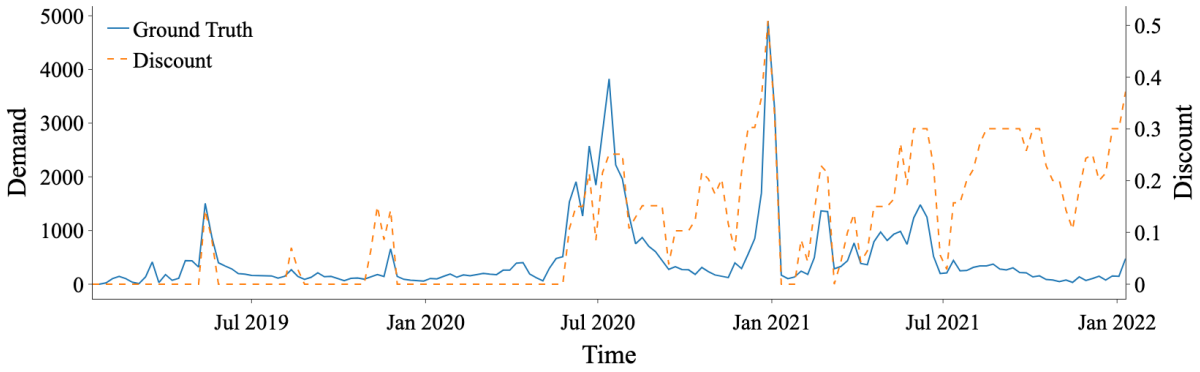


Figure 7: Demand and discount time series of an article, illustrating the positive correlation between discount (orange dashed) and demand (blue).

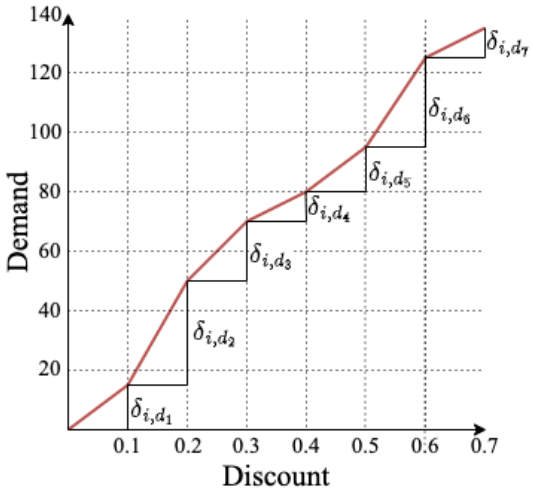


Figure 8: Piecewise linear monotonic demand response function as defined in (10), x-Axis: the discount level, y-axis: The increase in demand under the applied discount. The δ values represent the change in demand for a 10 percentage point change in discount.

3.2.6 Monotonic Demand Layer

The demand forecast model represents future demand as a function of future discount $d_{i,t+1:t+h}$. This requires not only to learn the temporally dependent dynamics of the demand time series but also the demand response function w.r.t. discount.

Figure 7 shows demand and discount time series, illustrating the positive correlation between the two quantities. Although it might be obvious that the demand always increases with an increase in discount, the underlying causal mechanism is subject to confounding, resulting in counter intuitive demand and discount curves. In order to address this issue, we model demand $q_{i,t+n}$ of future discount $d_{i,t+n}$ as a piece-wise linear and monotonically increasing demand response function in the monotonic demand layer.

The domain of the discount dependent function shown in Figure 8 is divided into segments of equal width. Each segment contains a linear function, specifying the increase in demand relative to the black price demand. We assume the following parameterization of the demand response function ξ:

$$\xi(d_{i,t+n}) = \hat{q}_{i,t+n} + \sigma_{i,t+n} \cdot \Big(\sum_{n=1}^{m} \delta_{i,d_n} + d_{i,t+n} \cdot \frac{\delta_{i,d_{m+1}}}{0.1} \Big) \qquad (10)$$
$$d_{i,t+n} \in [0, 0.7], m = \lfloor 10 \cdot d_{i,t+n} \rfloor$$

The monotonic demand layer parameterizes the demand function defined in (10) via two feed forward neural networks, ϕ_0 and ϕ_1

$$\begin{bmatrix} \delta_{id_1} \\ \delta_{id_2} \\ \vdots \\ \delta_{id_7} \end{bmatrix} = \phi_0(\gamma_{i,0:t}), \ \begin{bmatrix} \hat{q}_{i,t+n} \\ \sigma_{i,t+n} \end{bmatrix} = \phi_1(\kappa_{i,t+n}) \ . \qquad (11)$$

The input to ϕ_0 and ϕ_1 is the output of encoder and decoder, $\gamma_{i,0:t}$ and $\kappa_{i,t+n}$. The slopes $\delta_{id_1}, \ldots, \delta_{id_7}$ only depend on the encoder state, resulting in the same values for all future weeks. Black Price (see Table 1) demand $\hat{q}_{i,t+n}$ and scale parameter $\sigma_{i,t+n}$ are dependent on the decoder output column $\kappa_{i,t+n}$, indexed with $t+n$, similar as the future discount $d_{i,t+n}$. The Softplus[5] function ensures that the output of ϕ_0 and ϕ_1 results in a monotonically increasing demand function w.r.t. discount $d_{i,t+n}$ by construction. The monotonicity is a standard assumption in econometrics (see e.g., [38]). The final demand prediction at the future discount level $d_{i,t+n}$ is then the simple computation of (10), given the parameterization of ξ by ϕ_0 and ϕ_1. In order to provide demand predictions $q_{i,t+1:t+h}$ over the full forecast horizon $t+1 : t+h$, the steps are repeated for all future discounts $d_{i,t+1:t+h}$ and the corresponding decoder output columns in $\kappa_{i,t+1:t+h}$.

3.3 Near and Far Future Forecasts

In our approach, we decompose the forecast problem into near future and far future. The near future performance is of particular importance to our pricing use-case, hence we specialize our model architecture for this use case by having one decoder and one monotonic demand layer for each of the two forecast horizons. We set 5 week as the threshold for near future, and 5 to 20 week forecast will be considered as far future.

During training, we first train for seven epochs the decoder and monotonic demand layer for near future and freeze the decoder and demand layer for the far future. Then we freeze the components serving for near future and only train one final epoch for far future. We chose this specific training procedure to reduce training time, since the backpropagation of the error over the long forecast horizon turned out to be a significant cost factor. The near future loss only computes for the first 5 weeks, i.e. the near future horizon; the far future loss is covering all the 20 weeks. Note that the model is used to predict for 26 weeks ahead using the far future decoder and demand layer, depending on extrapolating beyond the training horizon.

In general, for forecasting tasks the further we forecast into the future the more deterioration there will be in the forecast. From Figure 9, we can see that the weighted ℓ_2 norm, which we use in practice as an evaluation metric, is much smaller on average in the near future part compared to the far future. The weighted ℓ_2 norm increases fast in the far future, which is expected. The far future is mainly used for long term effect and to estimate trend, therefore the accuracy requirement for far future is not as high as the near future.

3.4 Training

In our training procedure, we follow mostly standard deep learning approaches. For the loss function that we use to train our model, we note that in our down-stream price optimization applicate, we are primarily interested in maximizing expected profit and profit is a (mostly) linear function. The linearity of the profit function allows us to work directly with the mean of (1). Hence we use a loss function closely resembling the ℓ_2-loss:

$$\mathcal{L} = \sum_{j \in M} (v(\hat{q}_j) - v(q_j))^2 \qquad (12)$$

where q is the demand and $\hat{q}$ the predicted demand and $v(x) = 1 + x + x^2/2 + x^3/6$ is the third-order Taylor approximation of e^x. This showed a stabilizing effect in the beginning of the training since the model became less sensitive to large

[5]Smoothed out version of the ReLU [34] function.

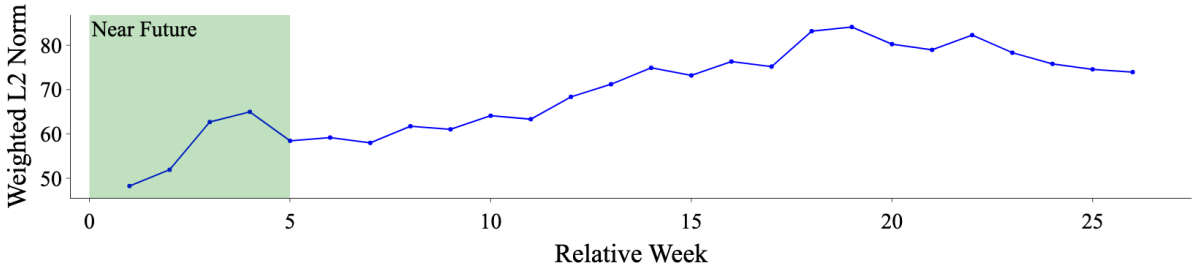


Figure 9: Demand model accuracy over a 26 weeks forecast horizon.

errors. Note that we need to exponentiate, since we apply Box-Cox log transformation to our targets, see Table 1. Index j ranges over all markets M. The loss is not computed on the observed sales but on the inferred demand (see Section 2.1). Recall from Section 3.3 that we place a special emphasis on the near future, since pricing decisions are updated frequently and only prices for the next time unit will materialize. We train the encoder and short-term decoder jointly for most of the epochs. In the last epochs of training, we freeze the encoder and only train the decoder for predicting the remaining far future weeks. This has the advantage to focus the encoder on the more important near-term forecasting problem but also has computational upsides as we do not need to train the high-dimensional decoder for the long-term.

3.5 Prediction

The demand forecast model generates article level demand predictions for 14 different countries in the price range of 0 to 70% discount in 5 percentage point discount steps. The output of this large scale prediction run is a so-called *demand grid* $\Upsilon \in R^{a \times c \times t \times d}$. The demand grid Υ spans over $t = 26$ future weeks, up to $a = 1 \times 10^6$ articles, $c = 14$ countries and $d = 15$ discount levels. All predictions are generated via the global demand forecasting model introduced in Section 3. In order to produce article specific predictions, the model first loads the specific features of one article like demand and price history as well as categorical features and other covariates 1. The model then predicts future demand for this specific article given the future discount level. Put differently, we answer the question: what would be the demand q of an article i in future week t if we keep everything impacting demand constant but only change the price of the article by applying discount d. This makes it inherently a causal forecasting [5] problem as described in (2). The demand grid results in a large data batch with up to 5.5×10^9 records and demand forecasts over the full range of future discount levels. Based on the discount dependent demand forecast grid, the down-stream price optimizer then selects an article specific future price trajectory that maximizes the achievable profit of the individual article in the bounds of certain business rules like stock levels and maximum discount steps.

4 Empirical Results

The model is trained on the full Zalando article assortment, processing 4800 samples per batch. We choose 6 attention layers each in the encoder and decoder. Each multi head attention layer contains $h = 23$ attention heads. Including the heads, the dimension of the Q, K and V tensors is $h \times (j + 1) \times d_k$ with $j = 51$[6] and $d_k = 4$[7]. For the monotonic demand layer, we choose seven segments of equal size 0.1. For the hyperparameter γ in the loss (12), we choose the demand share of the associated market. In total, the model contains approximately 5.3×10^7 trainable parameters.

A standard time to train the model is about 45 hours on a four GPU instance (ml.g4dn.12xlarge) at a cost of approximately 275 US$. We use a standard forecasting system set-up combining Spark and Amazon SageMaker [28, 8, 53], depicted in Figure 10. K8 refers to Kubernetes and S3 is the Amazon Web Service Storage Service.

4.1 Accuracy metrics

For measuring the accuracy of our demand forecast we need metrics that fulfill the following criteria:

1. comparable between datasets with different number of articles

[6] j is the observational time horizon and selected to cover a complete year starting at index 0, considering weekly time resolution.
[7] d_k is a hyperparameter defining the dimension of Q, K, V.

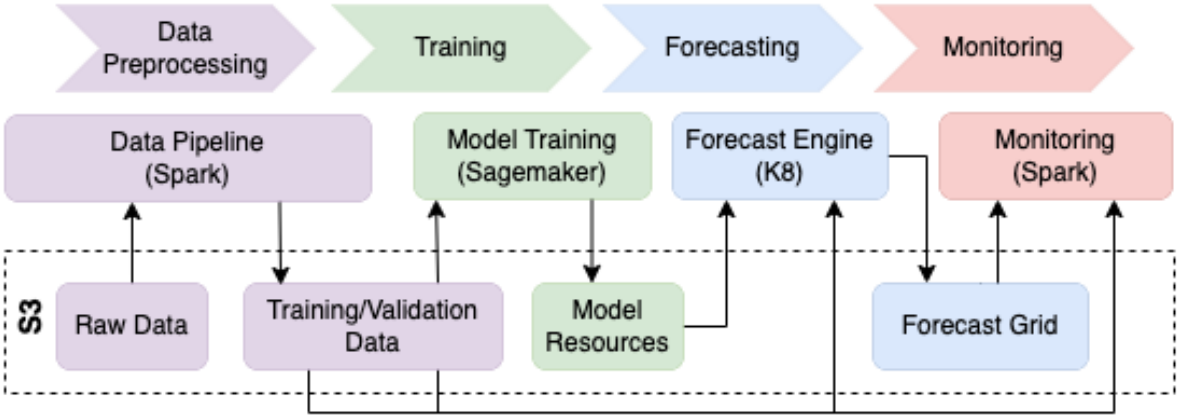


Figure 10: Forecasting Pipeline set-up in production.

2. weighted by how valuable an article is
3. working well with our downstream optimization task.

We define the *demand error*

$$\mathcal{D}_{T,h} = \sqrt{\frac{\sum_i \sum_{T=t+1}^{t+h} b_i (\hat{q}_{i,T} - q_{i,T})^2}{\sum_i \sum_{T=t+1}^{t+h} b_i q_{i,T}^2}} \tag{13}$$

and the *demand bias*

$$\mathcal{B}_{T,h} = \frac{\sum_i \sum_{T=t+1}^{t+h} b_i (\hat{q}_{i,T} - q_{i,T})}{\sum_i \sum_{T=t+1}^{t+h} b_i q_{i,T}} \tag{14}$$

as the custom metrics to assess the accuracy of our forecasting problem. Here, t is the last timepoint in the training set, h denotes the forecast horizon, $\hat{q}_{i,T}$ is the prediction for article i at timepoint T, $q_{i,T}$ is the corresponding true demand and b_i is the black price of article i.

The demand error $\mathcal{D}_{T,h}$ measures how large our forecasting error is on a single article level. We weight this by black price b_i to place a higher importance on more expensive articles as a proxy for the overall value for Zalando. As these typically generate more profit per sale, an error in the demand forecast for these articles is more costly for us. Scaling ensures that even in our setting, where the set of forecasted articles and sold changes from week to week, we obtain comparable values. In contrast, the demand bias indicates whether we are on average over or underpredicting while using the same weighting and scaling as the demand error.

We also experimented with (weighted and scaled) versions of more common accuracy metrics, like MAPE and MAE, and ran simulations how this changes affect the downstream optimization task. It turns out that outliers are an important factor for the performance so that a metric that resembles the RMSE more closely will have the best result with respect to the downstream pricing application. This feature is also the biggest drawback of our demand error metric as we observe that the error is largely driven by a small portion of the data. Especially forecast accuracy on articles with low and sparse demand (which is the majority of our articles) is not adequatly adressed by these metrics.

4.2 Model benchmarking

To illustrate accuracy of our transformer model we present a part of the empirical results we gathered. In particular we compare the model described above with a naive forecast that simply repeats the last observation, a tree-based model [20] using LightGBM [24] and the autoregressive recurrent neural network DeepAR [42] as implemented in GluonTS [1] for which we used the mean forecast as a point forecast. For the experiments, we chose 10 weeks in 2022, using all sales data from Germany from the category Textil. In Zalando this is the largest category, and performance here is highly indicative of the performance on the whole assortment. We focus primarily on the performance of the first weeks forecast. Table 2 shows the average metrics in the given weeks, along our own metrics defined as in Section 4.1, we also calculated more widely used metrics, namely RMSE and MAPE, where, to deal with zeros, we added 10 to both predicted and true demand values.

Forecaster	Demand Error		Demand Bias		RMSE		MAPE	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
Naive Forecaster	0.603	0.135	0.079	0.161	11.754	4.038	0.088	0.011
lightgbm	0.489	0.066	0.296	0.127	9.737	2.941	0.142	0.005
DeepAR	0.593	0.126	0.090	0.156	12.443	4.224	0.092	0.011
Transformer	**0.449**	0.04	**- 0.047**	0.064	**9.588**	2.725	**0.081**	0.007

Table 2: Model Performance

Comparison of our transformer models with commonly used alternative approaches and a naive baseline. The metrics are calculated over 10 different weeks in 2022 and measure the performance of the predictions for the first week. We report here the mean and standard deviation over the 10 weeks.

We observe that our transformer model performs best in terms of demand error and RMSE. While tree based models seem to be a close competitor in terms of demand error and were much faster to train, we saw a significantly higher bias. In terms of demand bias and MAPE, both Deep Learning models are very close to each other, but for RMSE again lightgbm and our transformer model have a clear advantage. It is worth mentioning that the naive baseline performs quite competitively on the one week forecast. A more detailed analysis reveals that especially for high selling articles using the naive forecast poses a strong baseline. Obviously during periods of high variation, as they happen at the beginning and the end of sales events, the naive forecast becomes almost useless. During the end of season sale in 2022 for example we observed a demand error of 0.877 for the naive forecaster while our transformer model reached a much more stable demand error of 0.491.

4.3 On the Benefits of Transformer-based Forecasting: First Results

In the overall literature on forecasting, conclusive evidence of outsized gains in predictive accuracy of transformer-based models is not yet available. On the one hand, results on publicly available data sets are positive [29], but not as overwhelmingly positive as in natural language processing (NLP) where transformers dominate. Careful comparisons with strong baselines (e.g., [7, 45]) place additional scrutiny on these results. We remark that transformer-based forecasting approaches tend to distinguish themselves by side-aspects such as long-term forecasting or interpretability whereas in NLP, the improvements in the core task are outsized.

On the other hand, on closed data sets, convincing evidence exists for the superiority of transformer-based forecasting models, e.g., [12, 52]. Our results above point into a similar direction. So, a natural question to ask is whether (and how) the success of transformer-based models can be associated to the nature of these closed source data sets and in particular the abundace of data available for companies like Zalando. To make progress in answering this question, we considered an often-cited reason for the success of transformers, so-called scaling laws (e.g., [23, 9]). These scaling laws state that the predictive performance on a test set continues to increase for transformers along various dimension including, for example, the size of the training set.

Concretely, we attempt to replicate scaling laws for transformers in forecasting, e.g., in [23, Figure 1] and whether we can observe level effects (see Figure 11).

Experimental Setup

- We chose three different forecast start dates distributed over 2021.
- We use 208 weeks of history to forecast 5 weeks for each each start date.
- On each of the three start dates, we use a representative subset of roughly 760K different articles and always use these as the reference point for testing with regards to the Demand Error. In this sense, we will have two layers of generalization present: The first stemming from predicting future demand from time series whose history is present in the training data and the second corresponding to completely unseen parts of the assortment.
- For smaller training sets we ensure that we always have a comparable number of batches per epoch (and hence, comparable training time) by iterating over the same time series several times.
- All other hyperparameters of the training procedure (batch size, cloud instance, learning parameters,...) remain constant and mimic our production setup.
- We included a naive forecast based on the last week's observation in the training horizon to check under which train dataset sizes our model starts to outperform.

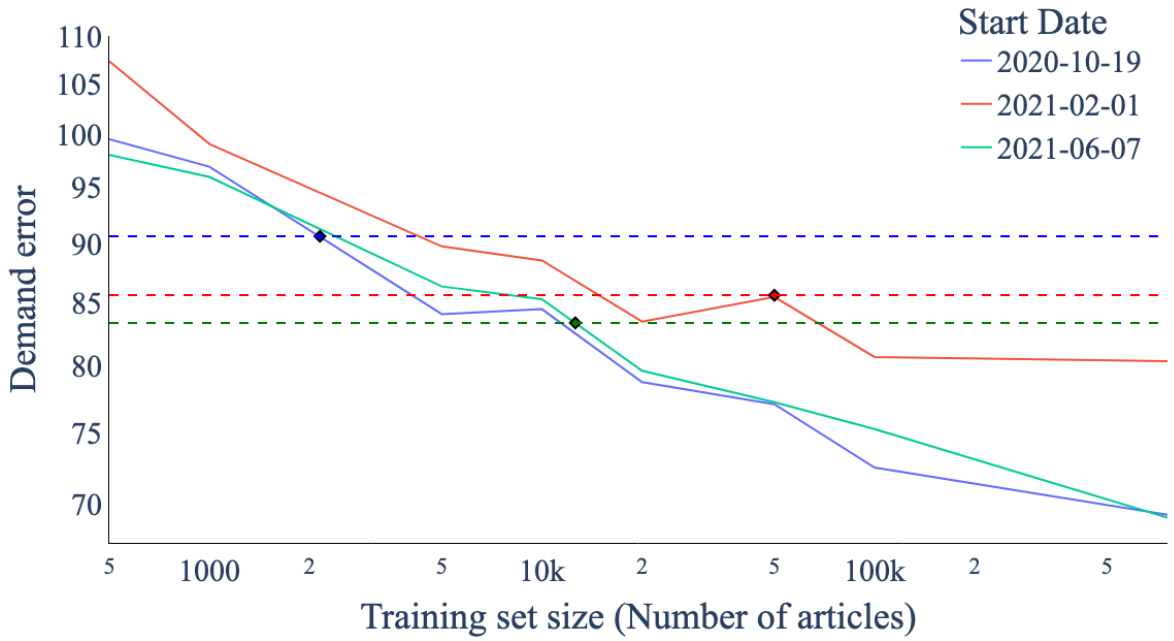


Figure 11: Log-log plot of the training-set-size-to-demand-error relationship for three different start dates with constant test set. The dashed lines represent the forecast performance of the naive forecaster (last observation in training horizon).

The demand error for each train dataset size (in terms of the number of time-series) as well as different start dates are shown in Fig 11. The naive forecast exhibits the expected behaviour and its performance is naturally constant in the size of the training set. Interestingly, the naive forecaster is already outperformed by training our model on a small fraction of the assortment (between 0.2% and 6.6% for the three start dates under consideration). For two out of the three start dates, the scaling law holds well. Similar scaling laws on public data sets in forecasting with other transformer-based models have not been reported so far (and rather the contrary on publicly available data sets [7, Figure 2]).

Clearly, the above experiment is only a first step in understanding a potential superiority of the transformer-based approach. Our data sets and associated training times are still humble compared to what state-of-the-art NLP-models use, so using more available data (e.g., by going further back in time) presents an obvious opportunity. We believe that this is the first time that evidence for scaling laws for transformers in forecasting was presented.

5 Related Work

Our demand model is a customized transformer-based forecasting model [50]. When we launched it in 2019, the current rich landscape of transformer based forecasting models was not yet available. So, we drew inspiration from advances in deep-learning architectures applied to sequential learning problems [46] and from the success of the transformer architecture in the field of natural language processing [11]. In this, our development is similar to other online retailers [12] who worked on transformer-based time series models concurrently to our work, but prior to the publicly available state of the art. For the latter, remarkable current transformer models include [29, 27, 54]. But, it remains an open question how well these fare against proven state of the art deep-learning based models (e.g., [42, 35]): We have not yet compared ourselves against these baselines for practical reasons and the introduction of state of the art open source libraries [1] will facilitate this. We refer to [4] for an overall overview on deep-learning based forecasting models.

Note that the approach described here is a point forecast. While clearly not best practice, the (piece-wise) linearity of the profit function used down-stream [26] partially remediates this methodological short-coming. We point to Gouttes et al. [16] how to turn our model into a flexibly parameterized probabilistic forecasting model. Other generic approaches amend themselves readily to our model such as conformal forecasting [44] or quantile-based approaches [15, 22]. We note further that we could handle the intermittency of the demand data more directly, e.g., [48, 21]. This is future work.

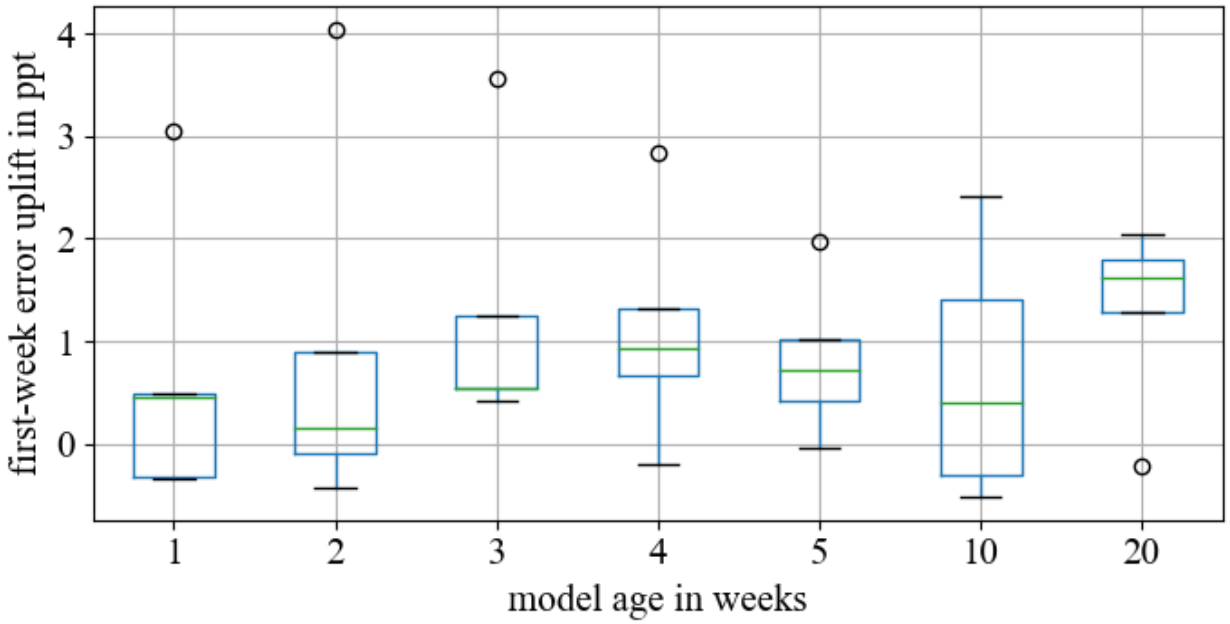


Figure 12: Effect of retraining on accuracy: The plot shows how much accuracy the model loses if not retrained regularly, provided a certain level of randomness introduced by the training process.

Modelling the demand response function as a piece-wise linear function is, to the best of our knowledge, novel. However, we remark that in other contexts, this has been proposed before in forecasting [15] and the general theme of modelling the monotonic relationship between price and demand is explored recently e.g., [30] although not for deep-learning based models.

Finally, we remark that our model (1) is an over simplification in the sense that we ignore cross-item relationships or the hierarchical relationship of our data. Recent work for multi-variate forecasting in a deep learning context (e.g., [41, 10, 40]) and hierarchical forecasting (e.g., [39, 47, 17]) exists, but we leave the application of it for future work.

6 Practical Considerations

We have so far described the data and the model assuming a mostly static setting. In reality however, forecasts need to be made available on a weekly schedule in an ever-changing environment. This leads to additional challenges which in turn requires a number of additional artefacts. We summarize this briefly on a high-level.

First, we must be able to judge whether our live forecasting accuracy is in line with the accuracy we have in our backtests. For this, we have extensive monitoring metrics that compare incoming live sales data with our forecasts and alarm us, if we observe a deterioration in practice. Second, we must be able to roll-out model changes (e.g., adjustments of hyper-parameters, additional covariates or library updates would be small examples). For this, a deployment pipeline is used with a large suite of test scenarios that cover aspects such as accuracy or training & inference speed in a variety of scenarios. Third, we monitor the input data distribution to scan it for distribution shifts. Note that we could use this for example for sophisticated re-training scheduling schemes, however, experiments reported in Figure 12 have shown that a weekly retraining scheme offers best results.

Finally, we note that the demand forecasting model described here, is only one input of more into a discount optimizer [26]. While our demand forecasts and price elasticities have some value in themselves, the most important artefact are discount decisions. We are only at the start of the journey to understand how forecast accuracy changes attribute to price changes in the complicated interplay between multiple models and (price) decisions that change the environment (and for which backtesting is hence hard). This is particularly involved because our systems evolves dynamically over the course of the seasonal cycle.

7 Conclusion

We have presented the demand forecasting model employed by Zalando SE, one of Europe's leading fashion online retailer. This global, transformer-based model delivers demand forecasts week in and week out since its initial inception in 2019, thereby predating some of the existing literature on transformer-based forecasting models by at least a year. We showed that this model is competitive against other methods. We also provide a first explanation of the superiority of transformer-based models by testing for scaling laws, as our closed data set contains millions of time series.

The incorporation of price or discount is the main modelling innovation that we presented here and it also presents the largest opportunity for future work. We are in fact most interested in causal or counterfactual forecasts – because these allow the most meaningful downstream decisions. Work on causal forecasting exists (e.g., [49, 31]), but is in its infancy. We believe that much further work is needed and hope that the exposition of our problem presented here will help to inspire such work further.

Acknowledgments

This article represents the effort of a larger group with everyone contributing to the manuscript and methodology in unique and impactful ways. We are grateful for their contributions without which we would not have this manuscript (and neither the model running in production at Zalando). We want to especially thank Tofigh Naghibi, who inspired the team to apply neural networks to our time series forecasting problem back in 2017. He did not only provide the initial idea but also contributed the first working prototype of the transformer based forecasting model. His work heavily influenced the team and was setting the foundation for a successful deep learning based forecasting application.

References

[1] Alexander Alexandrov, Konstantinos Benidis, Michael Bohlke-Schneider, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Danielle C Maddix, Syama Rangapuram, David Salinas, Jasper Schulz, et al. GluonTS: Probabilistic Time Series Models in Python. *JMLR*, 2019.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] Dzmitry Bahdanau, Kyungun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[4] Konstantinos Benidis, Syama Sundar Rangapuram, Valentin Flunkert, Yuyang Wang, Danielle Maddix, Caner Turkmen, Jan Gasthaus, Michael Bohlke-Schneider, David Salinas, Lorenzo Stella, François-Xavier Aubet, Laurent Callot, and Tim Januschowski. Deep learning for time series forecasting: Tutorial and literature survey. *ACM Comput. Surv.*, 55(6), dec 2022. ISSN 0360-0300. doi: 10.1145/3533382. URL https://doi.org/10.1145/3533382.

[5] Ioana Bica, Ahmed M Alaa, James Jordon, and Mihaela van der Schaar. Estimating counterfactual treatment outcomes over time through adversarially balanced representations. *arXiv preprint arXiv:2002.04083*, 2020.

[6] Michael Bohlke-Schneider, Shubham Kapoor, and Tim Januschowski. Resilient neural forecasting systems, 2022. URL https://arxiv.org/abs/2203.08492.

[7] Oliver Borchert, David Salinas, Valentin Flunkert, Tim Januschowski, and Stephan Günnemann. Multi-objective model selection for time series forecasting, 2022. URL https://arxiv.org/abs/2202.08485.

[8] Joos-Hendrik Böse, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Dustin Lange, David Salinas, Sebastian Schelter, Matthias Seeger, and Yuyang Wang. Probabilistic demand forecasting at scale. *Proc. VLDB Endow.*, 10 (12):1694–1705, aug 2017. ISSN 2150-8097. doi: 10.14778/3137765.3137775. URL https://doi.org/10.14778/3137765.3137775.

[9] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[10] Emmanuel de Bézenac, Syama Sundar Rangapuram, Konstantinos Benidis, Michael Bohlke-Schneider, Richard Kurle, Lorenzo Stella, Hilaf Hasson, Patrick Gallinari, and Tim Januschowski. Normalizing kalman filters for

multivariate time series analysis. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[12] Carson Eisenach, Yagna Patel, and Dhruv Madeka. MQTransformer: Multi-Horizon Forecasts with Context Dependent and Feedback-Aware Attention, 2020.

[13] Christos Faloutsos, Jan Gasthaus, Tim Januschowski, and Yuyang Wang. Classical and contemporary approaches to big time series forecasting. In *Proceedings of the 2019 International Conference on Management of Data*, SIGMOD '19, page 2042–2047, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450356435. doi: 10.1145/3299869.3314033. URL https://doi.org/10.1145/3299869.3314033.

[14] Robert Fildes, Shaohui Ma, and Stephan Kolassa. Retail forecasting: Research and practice. *International Journal of Forecasting*, 38(4):1283–1318, 2022. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2019.06.004. URL https://www.sciencedirect.com/science/article/pii/S016920701930192X. Special Issue: M5 competition.

[15] Jan Gasthaus, Konstantinos Benidis, Yuyang Wang, Syama Sundar Rangapuram, David Salinas, Valentin Flunkert, and Tim Januschowski. Probabilistic forecasting with spline quantile function rnns. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 1901–1910. PMLR, 16–18 Apr 2019. URL https://proceedings.mlr.press/v89/gasthaus19a.html.

[16] Adele Gouttes, Kashif Rasul, Mateusz Koren, Johannes Stephan, and Tofigh Naghibi. Probabilistic time series forecasting with implicit quantile networks, 2021.

[17] Xing Han, Sambarta Dasgupta, and Joydeep Ghosh. Simultaneously reconciled quantile forecasting of hierarchically related time series. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 190–198. PMLR, 13–15 Apr 2021.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[19] Tim Januschowski, Jan Gasthaus, Yuyang Wang, David Salinas, Valentin Flunkert, Michael Bohlke-Schneider, and Laurent Callot. Criteria for classifying forecasting methods. *International Journal of Forecasting*, 36 (1):167–177, 2020. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2019.05.008. URL https://www.sciencedirect.com/science/article/pii/S0169207019301529. M4 Competition.

[20] Tim Januschowski, Yuyang Wang, Kari Torkkola, Timo Erkkilä, Hilaf Hasson, and Jan Gasthaus. Forecasting with trees. *International Journal of Forecasting*, 38(4):1473–1481, 2022. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2021.10.004. URL https://www.sciencedirect.com/science/article/pii/S0169207021001679. Special Issue: M5 competition.

[21] Yunho Jeon and Sihyeon Seong. Robust recurrent network model for intermittent time-series forecasting. *International Journal of Forecasting*, 38(4):1415–1425, 2022. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2021.07.004. URL https://www.sciencedirect.com/science/article/pii/S0169207021001151. Special Issue: M5 competition.

[22] Kelvin Kan, François-Xavier Aubet, Tim Januschowski, Youngsuk Park, Konstantinos Benidis, Lars Ruthotto, and Jan Gasthaus. Multivariate quantile function forecaster. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera, editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 10603–10621. PMLR, 28–30 Mar 2022. URL https://proceedings.mlr.press/v151/kan22a.html.

[23] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020.

[24] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

[25] Nikolay Laptev, Jason Yosinsk, Li Li Erran, and Slawek Smyl. Time-series extreme event forecasting with neural networks at Uber. In *ICML Time Series Workshop*. 2017.

[26] Hanwei Li, David Simchi-Levi, Rui Sun, Michelle Xiao Wu, Vladimir Fux, Torsten J. Gellert, Thorsten Greiner, and Andrea Taverna. Large-scale price optimization for an online fashion retailer. *Social Science Research Network*, 2020.

[27] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/6775a0635c302542da2c32aa19d86be0-Paper.pdf.

[28] Edo Liberty, Zohar Karnin, Bing Xiang, Laurence Rouesnel, Baris Coskun, Ramesh Nallapati, Julio Delgado, Amir Sadoughi, Yury Astashonok, Piali Das, Can Balioglu, Saswata Chakravarty, Madhav Jha, Philip Gautier, David Arpin, Tim Januschowski, Valentin Flunkert, Yuyang Wang, Jan Gasthaus, Lorenzo Stella, Syama Rangapuram, David Salinas, Sebastian Schelter, and Alex Smola. Elastic machine learning algorithms in amazon sagemaker. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD '20, page 731–737, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450367356.

[29] Bryan Lim, Sercan Ö Arık, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764, 2021.

[30] Eleanor Loh, Jalaj Khandelwal, Brian Regan, and Duncan A. Little. Promotheus: An end-to-end machine learning framework for optimizing markdown in online fashion e-commerce. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, page 3447–3457, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450393850. doi: 10.1145/3534678.3539148. URL https://doi.org/10.1145/3534678.3539148.

[31] Valentyn Melnychuk, Dennis Frauen, and Stefan Feuerriegel. Causal transformer for estimating counterfactual outcomes. 2022. doi: 10.48550/ARXIV.2204.07258. URL https://arxiv.org/abs/2204.07258.

[32] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[33] Pablo Montero-Manso and Rob J. Hyndman. Principles and algorithms for forecasting groups of time series: Locality and globality. *International Journal of Forecasting*, 37(4):1632–1653, 2021. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2021.03.004. URL https://www.sciencedirect.com/science/article/pii/S0169207021000558.

[34] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[35] Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437*, 2019.

[36] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[37] Judea Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, 2nd edition, 2009.

[38] Robert L. Phillips. *Second Edition*. Stanford University Press, Redwood City, 2021. ISBN 9781503614260. doi: doi:10.1515/9781503614260. URL https://doi.org/10.1515/9781503614260.

[39] Syama Sundar Rangapuram, Lucien D Werner, Konstantinos Benidis, Pedro Mercado, Jan Gasthaus, and Tim Januschowski. End-to-end learning of coherent probabilistic forecasts for hierarchical time series. In *Proceedings of the 38th International Conference on Machine Learning*, pages 8832–8843, 2021.

[40] Kashif Rasul, Abdul-Saboor Sheikh, Ingmar Schuster, Urs Bergmann, and Roland Vollgraf. Multivariate probabilistic time series forecasting via conditioned normalizing flows, 2021.

[41] David Salinas, Michael Bohlke-Schneider, Laurent Callot, Roberto Medico, and Jan Gasthaus. High-dimensional multivariate forecasting with low-rank gaussian copula processes. *Advances in neural information processing systems*, 32:6827–6837, 2019.

[42] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191, 2020.

[43] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[44] Kamile Stankeviciute, Ahmed M. Alaa, and Mihaela van der Schaar. Conformal time-series forecasting. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 6216–6228. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/312f1ba2a72318edaaa995a67835fad5-Paper.pdf.

[45] Fan-Keng Sun and Duane S. Boning. Fredo: Frequency domain-based long-term time series forecasting, 2022. URL https://arxiv.org/abs/2205.12301.

[46] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[47] Filotas Theodosiou and Nikolaos Kourentzes. Forecasting with deep temporal hierarchies. *Available at SSRN: https://ssrn.com/abstract=3918315 or http://dx.doi.org/10.2139/ssrn.3918315*, 2021.

[48] Ali Caner Türkmen, Tim Januschowski, Yuyang Wang, and Ali Taylan Cemgil. Forecasting intermittent and sparse time series: A unified probabilistic framework via deep renewal processes. *PLOS ONE*, 16(11):1–26, 11 2021. doi: 10.1371/journal.pone.0259764. URL https://doi.org/10.1371/journal.pone.0259764.

[49] Leena Chennuru Vankadara, Philipp Michael Faller, Michaela Hardt, Lenon Minorics, Debarghya Ghoshdastidar, and Dominik Janzing. Causal forecasting:generalization bounds for autoregressive models, 2021. URL https://arxiv.org/abs/2111.09831.

[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[51] Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. A multi-horizon quantile recurrent forecaster, 2017. URL https://arxiv.org/abs/1711.11053.

[52] Sitan Yang, Carson Eisenach, and Dhruv Madeka. Mqretnn: Multi-horizon time series forecasting with retrieval augmentation, 2022. URL https://arxiv.org/abs/2207.10517.

[53] Matei Zaharia, Reynold S. Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J. Franklin, Ali Ghodsi, Joseph Gonzalez, Scott Shenker, and Ion Stoica. Apache Spark: A unified engine for big data processing. *Communications of the ACM*, 59(11):56–65, 2016.

[54] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting, 2020. URL https://arxiv.org/abs/2012.07436.