MQTransformer: Context Dependent Attention and Bregman Volatility

Kevin Chen [1] **Lee Dicker** [1] **Carson Eisenach** [1] **Dhruv Madeka** [1]

1. Introduction

Time series forecasting is a fundamental problem in machine learning with relevance to many applications including supply chain management, finance, healthcare, *etc*. As an example, consider a large e-commerce retailer with a system to produce forecasts of the demand distribution for a set of products at a target time T. Using these forecasts as an input, the retailer can then optimize buying and placement decisions. Accurate forecasts are important, but – perhaps less obviously – forecasts that don't exhibit excess volatility as a target date approaches minimize costly effects in a supply chain (6; 5).

Recent work applying deep learning to time-series forecasting focuses primarily on the use of recurrent and convolutional architectures (22; 34; 12; 21; 32); see Benidis et al. (4) for a complete overview. These are Seq2Seq architectures (26) – which consist of an *encoder* that summarizes an input sequence into a fixed-length context vector, and a *decoder* which produces an output sequence. Because real-world forecasting systems increasingly rely on neural nets, a need for black-box forecasting system diagnostics has arisen. The initial work in this area (27; 20; 28) considered the evolution of a sequence of forecasts for a single binary outcome in *continuous time*. More recently, (11) extended this work to quantile forecasts. In this paper, we develop a notion called *Bregman Volatility* to quantify the amount of forecast volatility beyond the forecast changes required to incorporate new information and improve accuracy. While Bregman Volatility can detect flaws in forecasts, how to incorporate that into model design is unexplored; existing multi-horizon forecasting architectures do not explicitly handle excess variation.

Another limitation in many existing architectures is the information bottleneck, where the encoder transmits information to the decoder via a single hidden state. Attention mechanisms (2) address this by allowing the decoder to take as input a weighted combination of relevant latent encoder states, rather than using a single context. Many variants have been proposed including self-attention and dot-product attention (19; 7; 30; 9), and transformer architectures (end-to-end attention with no recurrent layers) achieve state-of-the-art performance on NLP tasks. Absolute position encodings commonly used in the literature cannot be applied to time series forecasting. Our work differs from prior work on *relative position encodings* (8; 15; 25) in that we learn a representation from indicator variables of events relevant to the target application (e.g. holidays).

Summary of Contributions In this paper, we are concerned with both improving forecast accuracy *and* reducing excess forecast volatility. We present a set of novel architectures that seek to remedy some of inductive biases that are currently missing in state of the art MQ-Forecasters (32). Our main contributions are (1) positional encodings from event indicators, (2) horizon-specific decoder-encoder attention, (3) decoder self-attention for forecast evolution, (4) bregman volatility. We observe major increases in accuracy (5.5% in overall P90 quantile loss throughout the year, and **up to 33% during peak periods**) on our demand forecasting application. In terms of excess volatility, we see a reduction of 68% in Bregman Volatility for the mean forecast; we also evaluate using another, recent definition of excess volatility and see similar gains.

2. Methodology and Model Architecture

We consider the high-dimensional regression problem $p(y_{t+1,i}, \dots, y_{t+H,i} | \mathbf{y}_{:t,i}, \mathbf{x}^{(h)}_{:t,i}, \mathbf{x}^{(f)}_{:t,i}, \mathbf{x}^{(s)}_{i})$, where $y_{t+s,i}$, $\mathbf{y}_{:t,i}$, $\mathbf{x}^{(h)}_{:t,i}$, $\mathbf{x}^{(f)}_{t:,i}$, $\mathbf{x}^{(s)}_{i}$ denote future observations of the target time series i, observations of the target time series observed until time t, the past covariates, known future information, and static covariates, respectively. For sequence modeling problems, Seq2Seq (26) is the canonical deep learning framework and it has been adapted to time series forecasting (22; 34; 12; 21; 32; 24; 31).

The MQ-Forecaster framework (32) solves this by treating each series i as a sample from a joint stochastic process and feeding into a neural network which predicts Q quantiles for the next H periods. The learning objective is regression model to minimize the quantile loss, summed over all forecast creation times (FCTs) and quantiles $\sum_t \sum_q \sum_k L_q \left(y_{t+k}, \widehat{y}^{(q)}_{t+k} \right)$, where $L_q(y, \widehat{y}) = q(y - \widehat{y})_+ + (1-q)(\widehat{y} - y)_+$, $(\cdot)_+$ is the positive part operator, q is a quantile, and k is the horizon. We extend this family of models and for ease of exposition, we reformulate the generic probabilistic forecasting problem

[1]Forecasting Science, Amazon Inc., New York, NY, USA. Correspondence to: Carson Eisenach <ceisen@amazon.com>.

Continuous time methods for machine learning - ICML 2022, Baltimore, Maryland, USA, 2022.

as $p(y_{t+1,i}, \ldots, y_{t+H,i} | \mathbf{y}_{:t,i}, \mathbf{x}_{:t,i}, \mathbf{x}_i^{(l)}, \mathbf{x}^{(g)}, \mathbf{x}_i^{(s)})$, where $\mathbf{x}_{:t,i}$ are past observations of all covariates, $\mathbf{x}_i^{(l)} = \{\mathbf{x}_{s,i}^{(l)}\}_{s=1}^{\infty}$ are known covariates specific to time-series i, $\mathbf{x}^{(g)} = \{\mathbf{x}_s^{(g)}\}_{s=1}^{\infty}$ are the global, known covariates. Here known signifies that the model has access to (potentially noisy) observations of past and future values. This formulation is equivalent and known covariates can be included in the past covariates $\mathbf{x}_{:t}$. When it can be inferred from context, the time series index i is omitted.

The architecture is similar to MQ-(C)RNN (32), and consists of encoder, decoder and position encoding blocks. The position encoding output is a representation of global position information, $\mathbf{r}_t^{(g)} = \text{PE}_t^{(g)}(\mathbf{x}_i^{(g)})$, as well as time-series specific context information, $\mathbf{r}_t^{(l)} = \text{PE}_t^{(l)}(\mathbf{x}_i^{(l)})$. Intuitively, $\mathbf{r}_t^{(g)}$ captures position information that is independent of the time-series i (such as holidays), whereas $\mathbf{r}_t^{(l)}$ encodes time-series specific context information (such as promotions).The inputs are a time series of indicator variables *and require no feature-engineering or handcrafted functions*.

The encoder then summarizes past observations of the covariates into a sequence of hidden states $\mathbf{h}_t := \text{encoder}(\mathbf{y}_{:t}, \mathbf{x}_{:t}, \mathbf{r}_{:t}^{(g)}, \mathbf{r}_{:t}^{(l)}, \mathbf{s})$. Using these representations, the decoder produces an $H \times Q$ matrix of forecasts $\widehat{\mathbf{Y}}_t = \text{decoder}(\mathbf{h}_{:t}, \mathbf{r}^{(g)}, \mathbf{r}^{(l)})$. Note that in the decoder, the model has access to position encodings. Following the generic pattern given above, we present the MQTranformer architecture. First, define the combined position encoding as $\mathbf{r} := [\mathbf{r}^{(g)}; \mathbf{r}^{(l)}]$. In the encoder we use a stack of dilated temporal convolutions (29; 32) to encode historical time-series and a multi-layer perceptron to encode the static features.

Our decoder incorporates our horizon specific and decoder self-attention blocks, and consists of two branches. The first (global) branch summarizes the encoded representations into horizon-specific ($\mathbf{c}_{t,h}$) and horizon agnostic ($\mathbf{c}_t^a$) contexts. The output branch consists of a self-attention block followed by a local MLP, which produces outputs using the same weights for each horizon. For FCT t and horizon h, the output is given by $(\widehat{y}_{t+h}^1, \ldots, \widehat{y}_{t+h}^Q) = m_L(\mathbf{c}_t^a, \mathbf{c}_{t,h}, \widetilde{\mathbf{c}}_{t,h}, \mathbf{r}_{t+h})$. Next we describe the specifics of our position encoding and attention blocks.

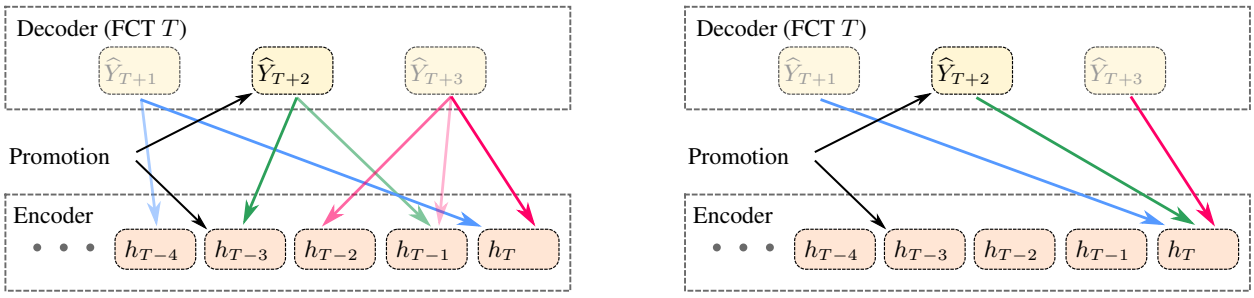
Table 1. Attention weight and output computations

BLOCK	ATTENTION WEIGHTS	OUTPUT
DECODER-ENCODER ATTENTION	$A_{t,s}^h = \mathbf{q}_t^{h,\top} \mathbf{W}_q^\top \mathbf{W}_k \mathbf{k}_s$ (1) $\mathbf{q}_t^h = [\mathbf{h}_t; \mathbf{r}_t; \mathbf{r}_{t+h}]$ $\mathbf{k}_s = [\mathbf{h}_s; \mathbf{r}_s]$ $\mathbf{v}_s = \mathbf{h}_s$	$\mathbf{c}_{t,h} = \sum_{s=t-L}^{t} A_{t,s}^h \mathbf{W}_v \mathbf{v}_s$ (2)
DECODER SELF-ATTENTION	$A_{t,s,r}^h = \mathbf{q}_{t,h}^\top \mathbf{W}_q^{h,\top} \mathbf{W}_k^h \mathbf{k}_{s,r}$ (3) $\mathbf{q}_{t,h} = [\mathbf{h}_t; \mathbf{c}_{t,h}; \mathbf{r}_t; \mathbf{r}_{t+h}]$ $\mathbf{k}_{s,r} = [\mathbf{c}_{s,r}; \mathbf{r}_s; \mathbf{r}_{s+r}]$ $\mathbf{v}_{s,r} = \mathbf{c}_{s,r}$	$\widetilde{\mathbf{c}}_{t,h} = \sum_{(s,r) \in \mathcal{H}(t,h)} A_{s,t,r}^h \mathbf{W}_v^h \mathbf{v}_{s,r}$, (4) $\mathcal{H}(t,h) := \{(s,r) \mid s + r = t + h\}$

Horizon-Specific Decoder-Encoder Attention To motivate the horizon-specific attention unit, consider a retail demand forecasting task. At each FCT T, the forecaster produces forecasts for multiple target horizons, which may contain different events such as promotions or holidays. Prior work (32) incorporated horizon-specific contexts with the functional form $c_{t,h} = f(h_t, r_{t+h})$ for each FCT, target horizon. Because events like promotions or holidays – which are strong predictors of observed demand – are fairly sparse, this functional form is insufficient. Instead, a function of the form $c_{t,h} = f(h_1, r_1, \ldots, h_t, r_t, r_{t+h})$ allows the model to use information from many past periods – see Figure 1.

Decoder Self-Attention Our Bregman Volatility theory below shows a deep connection between accuracy and volatility. We leverage this connection to develop a novel decoder self-attention scheme for multi-horizon forecasting. To motivate the development, consider a model which forecasts values of 40, 60 when the demand has constantly been 50 units. We would consider this model to have excess volatility. Similarly, a model forecasting 40, 60 when demand jumps between 40 and 60 units would not be considered to have excess volatility. The first model fails to learn from its past forecasts.

To ameliorate this, we pass information of the previous forecast errors to the current forecast. For each FCT t and horizon h, the model attends only on the previous forecasts using a query containing the demand information for that period. Formally, the attention scores are given by (10). The horizon-specific and feedback-aware outputs, $\widetilde{\mathbf{c}}_{t,h}$, are given by (11). Note how we sum only over previous forecasts of the same period. In Section 4 we demonstrate the importance of this choice by comparing to a more traditional decoder-self attention unit that attends over all past forecasts. See Figure 4 in Appendix B for an illustration.



(a) MQTransformer decoder at FCT T.

(b) MQCNN decoder at FCT T

Figure 1. Example of a demand forecasting task where periods $T-3$ and $T+2$ both have promotions. The encoded context h_{T-3}, the last time the item had a promotion, contains useful information for forecasting for target periods that also have a promotion.

3. Bregman Volatility

Prior work (11) proposed a notion of excess volatility that is defined in terms of the quadratic variation of a martingale derived from the forecasts as the target horizon approaches. Assume that some number $X \in \mathbb{R}$ (e.g. customer demand) is observed at a given time point in the future, T_∞, and forecasts for X, denoted $\widehat{X}_t$, are created at time points $t = 1, 2, \ldots, T < T_\infty$. The accuracy of each forecast $\widehat{X}_t$ is measured by $c(\widehat{X}_t - X)$, where $c : \mathbb{R} \to \mathbb{R}$ is some convex cost function that is minimized at 0. We assume that c is differentiable on all of $\mathbb{R}$, except possibly at 0. Let $\mathcal{F}_t$ denote the σ-field of information available at time t, so, in particular, $\mathcal{F}_t \subseteq \mathcal{F}_{t+1}$. Define the function $c_X : \mathbb{R} \to \mathbb{R}$ by $c_X(x) = c(x - X)$. Our goal is to find forecasts $\widehat{X}_t \in \mathcal{F}_t$ that are optimal in the sense that $\widehat{X}_t = \operatorname{argmin}_{X_t \in \mathcal{F}_t} \mathbb{E}[c_X(X_t) \mid \mathcal{F}_t]$. Observe that the loss function c determines the form of the corresponding forecast $\widehat{X}_t$. For example, if the cost function is squared-error loss, then the forecasts are conditional means, while if cost function is quantile loss, then the forecasts are conditional quantile forecasts.

By definition, the evolution of the sequence of forecasts $\widehat{X}_1, \widehat{X}_2, \ldots, \widehat{X}_T$ should satisfy properties determined by their Bregman volatility (see below). In particular, the forecast accuracy improvement from time $t = 1$ to time $t = T$ should equal the Bregman volatility of the sequence of forecasts, in expectation. To our knowledge, Bregman volatility is a newly defined concept. However, it builds on Bregman divergence, an important tool from convex analysis for measuring the distance between points with respect to a specified convex function (e.g. (3)).

Let $D_{c_X} : \mathbb{R}^2 \to \mathbb{R}$ denote the Bregman divergence of c_X, which is given by $D_{c_X}(x, y) = c_X(x) - c_X(y) - c_X'(y)(x - y)$, for $x, y \in \mathbb{R}$. We define the *t-th Bregman volatility* of a sequence of forecasts $\widehat{X}_1, \ldots, \widehat{X}_T$ to be $\mathrm{Vol}_{c_X}^{(t:T)} = \sum_{t'=t}^{T-1} D_{c_X}(\widehat{X}_{t'}, \widehat{X}_{t'+1})$. To connect Bregman volatility and forecast accuracy, observe that for $t < T$, $\mathbb{E}[c_X(\widehat{X}_t)] = \mathbb{E}[c_X(\widehat{X}_T)] + \sum_{t'=t}^{T-1} \mathbb{E}[c_X'(\widehat{X}_{t'+1})(\widehat{X}_{t'} - \widehat{X}_{t'+1})] + \mathbb{E}[\mathrm{Vol}_{c_X}^{(t:T)}]$.

Theorem 3.1. *Let $c : \mathbb{R} \to \mathbb{R}$ be a convex function minimized at 0, which is differentiable everywhere except possibly at 0. Assume that the sequence of forecasts $\widehat{X}_1, \ldots \widehat{X}_T$ satisfies $\widehat{X}_t = \operatorname{argmin}_{X_t \in \mathcal{F}_t} \mathbb{E}[c_X(X_t) \mid \mathcal{F}_t]$. If c is differentiable at 0 or if the distribution $X \mid \mathcal{F}_t$ is continuous with a differentiable probability density function, then $\mathbb{E}[\mathrm{Vol}_{c_X}^{(t:T)}] = \mathbb{E}[c_X(\widehat{X}_t)] - \mathbb{E}[c_X(\widehat{X}_T)]$.*

In other words, Theorem 3.1 says that the expected Bregman Volatility is equal to the expected accuracy gain. To operationalize this result, we point out that if $\mathrm{Vol}_{c_X}^{(t:T)}$ systematically deviates from $c_X(\widehat{X}_t) - c_X(\widehat{X}_T)$, then this indicates that the forecasts are *not* optimal and can potentially be improved. Theorem 3.1 covers the case where c is differentiable at zero or $X \mid \mathcal{F}_t$ is continuous, but can be extended to non-differentiable c and discrete X.

3.1. Forecast Regression - A Procedure to Improve Accuracy and Volatility

The previous section gives us a metric, the Bregman volatility, which can be used to diagnose forecast optimality (i.e. for an optimal forecast, the Bregman volatility should equal the accuracy gain). In this section, we present a procedure – *Forecast Regression* – for adjusting the forecast $\widehat{X}_{t+1}$ to account for the previous forecasts $\widehat{X}_1, \ldots, \widehat{X}_t$.

Theorem 3.2. *Suppose we have a sequence of adjusted forecasts $\widetilde{X}_1, \ldots, \widetilde{X}_T$ defined by $\widetilde{X}_1 = \widehat{X}_1$ and $\widetilde{X}_t = \sum_{t'=1}^{t} \beta_{t',t} \widehat{X}_{t'}$, where $(\beta_{1,t}, \ldots, \beta_{t,t}) = \boldsymbol{b}_t := \operatorname{argmin}_{\boldsymbol{b} \in \mathbb{R}^t} \mathbb{E}\left[c_X\left(\sum_{t'=1}^{t} \beta_{t',t} \widehat{X}_{t'}\right) \mid \mathcal{F}_t\right]$ for $1 < t \leq T$. Then $E[c_X'(\widehat{X}_{t'+1})(\widehat{X}_{t'} - \widehat{X}_{t'+1})] = 0$ and $E[\mathrm{Vol}_{c_X}^{(t:T)}] = \mathbb{E}[c_X(\widehat{X}_t)] - \mathbb{E}[c_X(\widehat{X}_T)]$, with $\widetilde{X}_{t'}, \widetilde{X}_{t'+1}$ in place of $\widehat{X}_{t'}, \widehat{X}_{t'+1}$ and $\sum_{t=1}^{T} \mathbb{E}[c_X(\widetilde{X}_t)] \leq \sum_{t=1}^{T} \mathbb{E}[c_X(\widehat{X}_t)]$.*

Theorem 3.2 follows immediately from the definition of $\boldsymbol{b}_t$ and implies that linear regression can improve both forecast accuracy and volatility. For quantile loss, Forecast Regression can still be utilized to improve accuracy and Bregman Volatility, but the derivation requires replacing some equalities with inequalities as quantile loss is non-differentiable.

3.2. Decoder Self-Attention Optimizes Bregman Vol

Here we justify the claim that the Decoder Self-Attention module in MQ-Transformer optimizes Bregman Volatility. Specifically, we show that the Decoder Self-Attention module is a generalization of the Forecast Regresson procedure. We fix a specific FTD, so that all forecasts and demand are for that FTD and for clarity we drop references to the forecasting horizon and position encoding. We represent the query and key using f_q and f_k to represent general non-linear functions computed by neural nets. Then the query is $f_q(\text{forecast}, \text{demand})$ and the key is $f_k(\text{forecast})$.

We write the forecasts in the same notation as Theorem 3.2, i.e. $\widehat{X}_{t'}$ is the forecast produced by MQ-Transformer *before* the Decoder Self-Attention module. It is not required for the Bregman Volatility results to use the previous demand so we write a simplified Decoder Self-Attention Weight matrix without the demand term as $A^{simple}_{t,t'} = f_{q,t}(\widehat{X}_t)^T W_q^T W_k f_{k,t'}(\widehat{X}_{t'})$, where $A^{simple}_{t,t'}$ is the attention weight between weeks t and t' in the output. The output of the Decoder Self-Attention block is a set of adjusted forecasts $\widetilde{X}_t = \sum_{t' \le t} A^{simple}_{t',t} \widehat{X}_{t'}$. Rewriting the simplified Decoder Self-Attention optimization problem in the notation of Theorem 3.2, we have $\boldsymbol{b}_t = \text{argmin}_{\boldsymbol{b} \in \mathbb{R}^N} \mathbb{E}\big[c_{QL}\big(\sum_{t'=1}^{t} A^{simple}_{t',t}(\boldsymbol{b})\widehat{X}_{t'}\big) \mid \mathcal{F}_t\big]$. Here we write the Attention matrix $A^{simple}_{t',t}$ as $A^{simple}_{t',t}(\boldsymbol{b})$ to emphasize the Attention matrix is computed by a neural net with a large number N of parameters, and we reinterpret the $\boldsymbol{b}$ coefficients as neural net weights of MQ-Transformer. Decoder Self-Attention thus generalizes the Forecast Regression procedure and therefore optimizes Bregman Volatility.

4. Empirical Results

We evaluate our architecture on a demand forecasting problem for a large-scale e-commerce retailer with the objective of producing multi-horizon forecasts for the next 52 weeks. We conduct our experiments on a set of 2 million products in the US store. Each model is trained on three years of data (2015-2018); one year (2018-2019) is held out for back-testing. To assess the effects of each innovation, we ablate by removing components one at a time. The architectures we compare are the baseline (MQ-CNN), MQTransformer (MQT), MQTransformer without decoder self-attention (MQT-NoDS), and MQ-Transformer with a decoder self-attention unit that attends over all past forecasts (MQT-All). MQ-CNN is selected as the baseline since prior work demonstrate that MQ-CNN outperforms MQ-RNN and DeepAR on this dataset (32). Results on public datasets can be found in Appendix D and full results on the demand forecasting task can be found in Appendix C.

Forecast Accuracy Table 2 summarizes several key metrics that demonstrate the accuracy improvements achieved by adding our proposed attention mechanisms to the MQ-CNN architecture. We consider overall quantile loss, as well as quantile loss for specific target periods (seasonal peaks, promotions). We also measure post-peak ramp-down performance – models that suffer issues with alignment will continue to forecast high for target weeks after a seasonal peak. By including MQTransformer's attention mechanisms, we see up to 33% improvements for seasonal peaks and 24.9% improvements on promotions versus MQ-CNN. Further, the ablation analysis (against MQT-NoDS and MQT-All) shows that both novel attention units are important to improving accuracy, and that our decoder-self attention scheme (where we attend only over past forecasts for the same target period) introduces a powerful inductive bias.

Forecast Volatility Table 2 also shows the excess bregman volatility of each architecture, on the same forecasting problem when ℓ_2 loss is used for the mean forecast. Evaluation using Thereshold Volatility – a related notion from (11) – on the quantile forecasts yields a similar result (see Appendix C). We conclude our decoder self-attention is critical to reducing excess volatility.

Table 2. P50 (50th percentile) and P90 (90th percentile) quantile loss on the backtest year along different dimensions. Values indicate relative performance versus the baseline. E.V. denotes excess Bregman volatility, rescaled versus the baseline. Lower is better.

	MQ-CNN			MQT			MQT-NoDS			MQT-All		
Dimension	P50	P90	EV	P50	P90	EV	P50	P90	EV	P50	P90	EV
Overall	1.000	1.000	1.000	**0.977**	**0.945**	**0.320**	0.983	0.963	2.020	**0.977**	0.957	0.630
Seasonal Peak 1	1.000	1.000	–	**0.865**	**0.667**	–	0.881	0.688	–	0.908	0.729	–
Seasonal Peak 2	1.000	1.000	–	0.957	**0.884**	–	0.984	0.946	–	**0.953**	0.931	–
Peak Rampdown	1.000	1.000	–	0.987	**0.978**	–	**0.985**	0.997	–	0.990	0.991	–
Promotion Type 1	1.000	1.000	–	**0.945**	**0.800**	–	0.969	0.824	–	0.953	0.842	–
Promotion Type 2	1.000	1.000	–	**0.868**	**0.751**	–	0.895	0.762	–	0.914	0.808	–

References

[1] AUGENBLICK, N. and RABIN, M. (2019). Belief movement, uncertainty reduction, and rational updating. Tech. rep., Haas School of Business, University of California, Berkeley.

[2] BAHDANAU, D., CHO, K. and BENGIO, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv:1409.0473.

[3] BANERJEE, A., MERUGU, S., DHILLON, I. S., GHOSH, J. and LAFFERTY, J. (2005). Clustering with Bregman divergences. *Journal of Machine Learning Research* **6**.

[4] BENIDIS, K., RANGAPURAM, S. S., FLUNKERT, V., WANG, B., MADDIX, D., TURKMEN, C., GASTHAUS, J., BOHLKE-SCHNEIDER, M., SALINAS, D., STELLA, L. ET AL. (2020). Neural forecasting: Introduction and literature overview. arXiv:2004.10240.

[5] BRAY, R. L. and MENDELSON, H. (2012). Information Transmission and the Bullwhip Effect: An Empirical Investigation. *Management Science* **58** 860–875.

[6] CHEN, F., DREZNER, Z., RYAN, J. K. and SIMCHI-LEVI, D. (2000). Quantifying the Bullwhip Effect in a Simple Supply Chain: The Impact of Forecasting, Lead Times, and Information. *Management Science* **46** 436–443.

[7] CHENG, J., DONG, L. and LAPATA, M. (2016). Long short-term memory-networks for machine reading. arXiv:1601.06733.

[8] DAI, Z., YANG, Z., YANG, Y., CARBONELL, J., LE, Q. V. and SALAKHUTDINOV, R. (2019). Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context. In *ACL*.

[9] DEVLIN, J., CHANG, M.-W., LEE, K. and TOUTANOVA, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*.

[10] DUA, D. and GRAFF, C. (2017). UCI machine learning repository.
URL http://archive.ics.uci.edu/ml

[11] FOSTER, D. and STINE, R. (2021). Threshold Martingales and the Evolution of Forecasts. arXiv:2105.06834.

[12] GASPARIN, A., LUKOVIC, S. and ALIPPI, C. (2019). Deep Learning for Time Series Forecasting: The Electric Load Case. arXiv:1907.09207.

[13] GERD HEBER, N. S., ASGER LUNDE and SHEPPARD, K. K. (2009). Oxford-man institute's realized library.

[14] HEATH, D. C. and JACKSON, P. L. (1994). Modeling the evolution of demand forecasts with application to safety stock analysis in production/distribution systems. *IIE transactions* **26** 17–30.

[15] HUANG, C.-Z. A., VASWANI, A., USZKOREIT, J., SHAZEER, N., SIMON, I., HAWTHORNE, C., DAI, A. M., HOFFMAN, M. D., DINCULESCU, M. and ECK, D. (2018). Music Transformer: Generating Music with Long-Term Structure. arXiv:1809.04281.

[16] KINGMA, D. P. and BA, J. (2015). Adam: A Method for Stochastic Optimization. In *ICLR*.

[17] LI, S., JIN, X., XUAN, Y., ZHOU, X., CHEN, W., WANG, Y.-X. and YAN., X. (2019). Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In *NIPS*.

[18] LIM, B., ARIK, S. O., LOEFF, N. and PFISTER, T. (2019). Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting. arXiv:1912.09363.

[19] LUONG, M.-T., PHAM, H. and MANNING, C. D. (2015). Effective approaches to attention-based neural machine translation. arXiv:1508.04025.

[20] MADEKA, D. (2017). Accurate Prediction of Electoral Outcomes. arXiv:1704.02664.

[21] MUKHOTY, B. P., MAURYA, V. and SHUKLA, S. K. (2019). Sequence to sequence deep learning models for solar irradiation forecasting. In *IEEE Milan PowerTech*.

[22] NASCIMENTO, R. C., SOUTO, Y. M., OGASAWARA, E., PORTO, F. and BEZERRA, E. (2019). STConvS2S: Spatiotemporal Convolutional Sequence to Sequence Network for weather forecasting. arXiv:1912.00134.

[23] OSTROVSKI, G., DABNEY, W. and MUNOS, R. (2018). Autoregressive quantile networks for generative modeling. In *International Conference on Machine Learning*. PMLR.

[24] SALINAS, D., FLUNKERT, V., GASTHAUS, J. and JANUSCHOWSKI, T. (2020). Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting* **36** 1181–1191.

[25] SHAW, P., USZKOREIT, J. and VASWANI, A. (2018). Self-Attention with Relative Position Representations. In *NAACL*.

[26] SUTSKEVER, I., VINYALS, O. and LE, Q. V. (2014). Sequence to sequence learning with neural networks. In *NIPS*.

[27] TALEB, N. N. (2018). Election predictions as martingales: an arbitrage approach. *Quantitative Finance* **18** 1–5.

[28] TALEB, N. N. and MADEKA, D. (2019). All roads lead to quantitative finance. *Quantitative Finance* **19** 1775–1776.

[29] VAN DEN OORD, A., DIELEMAN, S., ZEN, H., SIMONYAN, K., VINYALS, O., GRAVES, A., KALCHBRENNER, N., SENIOR, A. and KAVUKCUOGLU, K. (2016). Wavenet: A generative model for raw audio. arXiv:1609.03499.

[30] VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, L., and POLOSUKHIN, I. (2017). Attention is all you need. In *NIPS*.

[31] WEN, R. and TORKKOLA, K. (2019). Deep Generative Quantile-Copula Models for Probabilistic Forecasting. In *ICML Time Series Workshop*.

[32] WEN, R., TORKKOLA, K., NARAYANASWAMY, B. and MADEKA, D. (2017). A multi-horizon quantile recurrent forecaster. In *NIPS Time Series Workshop*.

[33] WILLIAMS, D. (1991). *Probability with Martingales*. Cambridge University Press.

[34] YU, R., ZHENG, S., ANANDKUMAR, A. and YUE, Y. (2017). Long-term Forecasting using Higher Order Tensor RNNs. arXiv:1711.00073.

A. Additional Background and Related Work

A.1. Attention Mechanisms

Attention mechanisms can be viewed as a form of content based addressing, that computes an alignment between a set of *queries* and *keys* to extract a *value*. Formally, let $\mathbf{q}_1, \ldots, \mathbf{q}_t$, $\mathbf{k}_1, \ldots, \mathbf{k}_t$ and $\mathbf{v}_1, \ldots, \mathbf{v}_t$ be a series of queries, keys and values, respectively. The s^{th} attended value is defined as $\mathbf{c}_s = \sum_{i=1}^{t} \text{score}(\mathbf{q}_s, \mathbf{k}_t)\mathbf{v}_t$, where score is a scoring function – commonly $\text{score}(\mathbf{u}, \mathbf{v}) := \mathbf{u}^\top \mathbf{v}$. In the vanilla transformer model, $\mathbf{q}_s = \mathbf{k}_s = \mathbf{v}_s = \mathbf{h}_s$, where $\mathbf{h}_s$ is the hidden state at time s. Because attention mechanisms have no concept of absolute or relative position, some sort of position information must be provided. Vaswani et al. (30) uses a sinusoidal positional encoding added to the input to an attention block, providing each token's position in the input time series.

In the vanilla transformer (30), a sinusoidal position embedding is added to the network input and each encoder layer consists of a multi-headed attention block followed by a feed-forward sub-layer. For each head i, the attention score between query q_s and key k_t is defined as follows for the input layer

$$A^h_{s,t} = (\mathbf{x}_s + \mathbf{r}_s)^\top \mathbf{W}^{h,\top}_q \mathbf{W}^h_k (\mathbf{x}_t + \mathbf{r}_t) \tag{5}$$

where $\mathbf{x}_s$, $\mathbf{r}_s$ are the observation of the time series and the position encoding, respectively, at time s. In Section 2 we introduced attention mechanisms that differ in their treatment of the position dependent biases

A.2. Threshold Volatility

Prior work (11) describes an alternative notion of forecast volatility called Threshold Volatility and we include results for that notion here. Originally the *martingale model of forecast evolution* (MMFE) was conceived as a way to simulate demand forecasts used in inventory planning problems (14). Denoting by $\widehat{Y}_{T|t}$ the forecast for Y_T made at time $t \leq T$, the MMFE assumes that the forecast process $\{\widehat{Y}_{T|t}\}_t$ is martingale. Informally, a martingale captures the notion that a forecast should use all information available to the forecasting system at time t. Mathematically, a discrete time martingale is a stochastic process $\{X_t\}$ such that

$$\mathbb{E}[X_{t+1}|X_t, \ldots, X_1] = X_t.$$

We assume a working knowledge of martingales and direct the reader to Williams (33) for a thorough coverage in discrete time.

Taleb (27) describe how martingale forecasts correspond to rational updating, then expanded by Augenblick and Rabin (1). Taleb (27), Taleb and Madeka (28) and Augenblick and Rabin (1) go on to develop tests for forecasts that rule out martingality and indicate irrational or predictable updating for binary bets. Foster and Stine (11) further extend these ideas to quantile forecasts. They consider the coverage probability process $p_t := \mathbb{P}[Y_T \leq \tau | Y_s, s \leq t] = \mathbb{E}[I(Y_T \leq \tau)|Y_s, s \leq t]$, where τ denotes the forecast announced in the first period $t = 0$. Because $\{p_t\}$ is also a martingale, the authors show that $\mathbb{E}[(p_T - \pi)^2] = \sum_{t=1}^{T} \mathbb{E}[(p_t - p_{t-1})^2] = \pi(1-\pi)$, where $\pi = p_0$ is the expected value of p_T, a Bernoulli random variable, across realizations of the coverage process. In the context of quantile forecasting, π is simply the quantile forecasted.

The measure of excess volatility proposed is the quadratic variation process associated with $\{p_t\}$, $Q_s := \sum_{t=0}^{s}(p_t - p_{t-1})^2$. While this process is not a martingale, we do know that under the MMFE assumption, $\mathbb{E}[Q_T] = \pi(1-\pi)$. The martingale $V_t := Q_t - (p_t - \pi)^2$ is obtained from this process in the usual way by subtracting the compensator. In Appendix C we leverage the properties of $\{V_t\}$ to compare the dynamics of forecasts produced by a variety of models, demonstrating that our feedback-aware decoder self-attention units reduce excess forecast volatility.

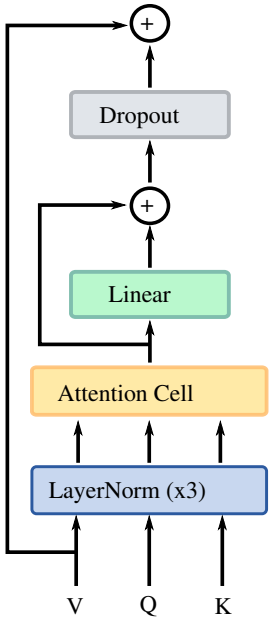
B. MQTransformer Architecture Details

Here we describe in detail the layers in our MQTransformer architecture, which is based off of the MQ-Forecaster framework (32) and uses a wavenet encoder (29) for time-series covariates. Before describing the layers in each component, Figure 3 outlines the MQTransformer architecture. On different datasets, we consider the following variations: choice of encoding for categorical variables, a tunable parameter d_h (dimension of hidden layers), dropout rate p_{drop}, a list of dilation rates for the wavenet encoder, and a list of dilation rates for the position encoding. The ReLU activation function is used throughout the network.

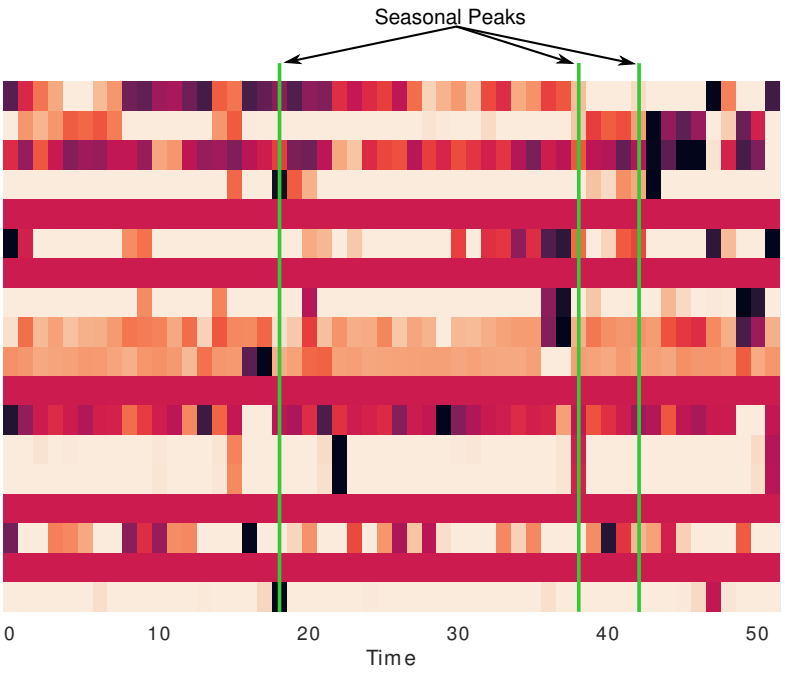
B.1. Input Embeddings and Position Encoding

Static categorical variables are encoded using either one-hot encoding or an embedding layer. Time-series categorical variables are one-hot encoded, and then passed through a single feed-forward layer of dimension d_h.

The global position encoding module takes as input the known time-series covariates, and consist of a stack of dilated, bi-directional 1-D convolution layers with d_h filters. After each convolution is a ReLU activation, followed by a dropout



(a) Basic structure of our time-series attention blocks



(b) Position encoding learned from event indicators in the large scale demand forecasting task (Section 4)

Figure 2. Components of the MQTransformer architecture

layer with rate p_{drop}, and the local position encoding is implemented as a single dense layer of dimension d_h. Figure 2b shows an example of the position encoding from a model trained on the large-scale demand forecasting task. See Table 3 for a high-level description of the network.

Table 3. MQTransformer encoder and decoder

ENCODER	
$\mathbf{h}_t^1 = \textsc{TemporalConv}(\mathbf{y}_{:t}, \mathbf{x}_{:t}, \mathbf{r}_{:t})$	(6)
$\mathbf{h}_t^2 = \textsc{FeedForward}(\mathbf{s})$	
$\mathbf{h}_t = [\mathbf{h}_t^1; \mathbf{h}_t^2]$,	
DECODER CONTEXTS	
$\mathbf{c}_{t,h} = \text{HSATTENTION}(\mathbf{h}_{:t}, \mathbf{r})$	(7)
$\mathbf{c}_t^a = \textsc{FeedForward}(\mathbf{h}_t, \mathbf{r})$	
$\mathbf{c}_t = [\mathbf{c}_{t,1}; \cdots; \mathbf{c}_{t,H}; \mathbf{c}_t^a]$	
$\widetilde{\mathbf{c}}_{t,h} = \text{DSATTENTION}(\mathbf{c}_{:t}, \mathbf{h}_{:t}, \mathbf{r})$,	

B.2. Encoder

After categorical encodings are applied, the inputs are passed through the encoder block. The encoder consists of two components: a single dense layer to encode the static features, and a stack of dilated, temporal convolutions. The time-series covariates are concatenated with the position encoding to form the input to the convolution stack. The output of the encoder block is produced by replicating the encoded static features across all time steps and concatenating with the output of the convolution.

B.3. Decoder

Please see Table 3 for a description of the blocks in the decoder. The dimension of each head in both the horizon-specific and decoder self-attention blocks is $d_h/2$. The dense layer used to compute $\boldsymbol{c}_t^a$ has dimension $d_h/2$. The output block is two

layer MLP with hidden layer dimension $d_h/2$, and weights are shared across all time points and horizons. The output layer has one output per horizon, quantile pair. Figure 2a shows the structure of both our attention blocks, where the sub-layer "Attention Cell" is replaced with either the decoder-encoder or decoder-self attention.

Table 4. Attention weight and output computations for blocks introduced in Section 2

BLOCK	ATTENTION WEIGHTS		OUTPUT	
DECODER-ENCODER ATTENTION	$A^h_{t,s} = \mathbf{q}^{h,\top}_t \mathbf{W}^\top_q \mathbf{W}_k \mathbf{k}_s$ $\mathbf{q}^h_t = [\mathbf{h}_t; \mathbf{r}_t; \mathbf{r}_{t+h}]$ $\mathbf{k}_s = [\mathbf{h}_s; \mathbf{r}_s]$ $\mathbf{v}_s = \mathbf{h}_s$	(8)	$\mathbf{c}_{t,h} = \sum_{s=t-L}^{t} A^h_{t,s} \mathbf{W}_v \mathbf{v}_s$	(9)
DECODER SELF-ATTENTION	$A^h_{t,s,r} = \mathbf{q}^\top_{t,h} \mathbf{W}^{h,\top}_q \mathbf{W}^h_k \mathbf{k}_{s,r}$ $\mathbf{q}_{t,h} = [\mathbf{h}_t; \mathbf{c}_{t,h}; \mathbf{r}_t; \mathbf{r}_{t+h}]$ $\mathbf{k}_{s,r} = [\mathbf{c}_{s,r}; \mathbf{r}_s; \mathbf{r}_{s+r}]$ $\mathbf{v}_{s,r} = \mathbf{c}_{s,r}$	(10)	$\widetilde{\mathbf{c}}_{t,h} = \sum_{(s,r)\in\mathcal{H}(t,h)} A^h_{s,t,r} \mathbf{W}^h_v \mathbf{v}_{s,r},$ $\mathcal{H}(t,h) := \{(s,r) \mid s + r = t + h\}$	(11)

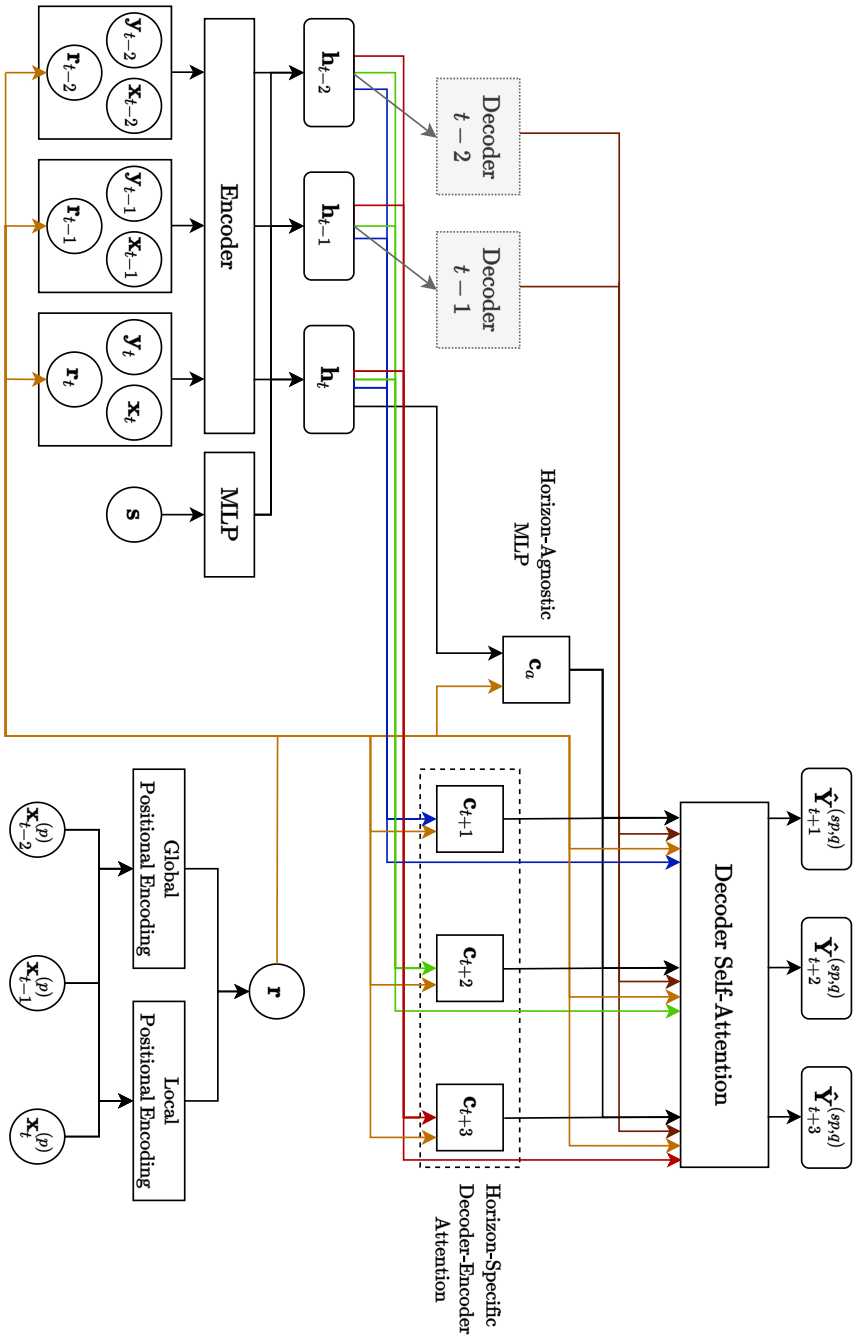
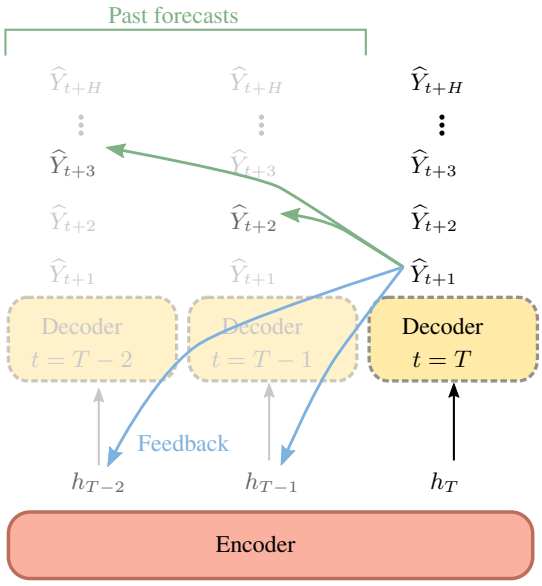
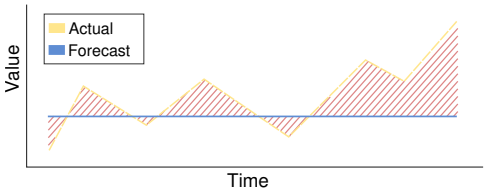


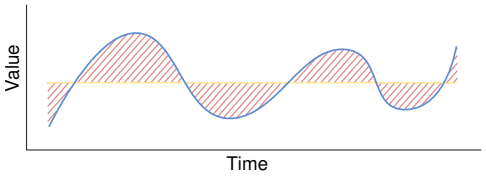
Figure 3. MQTransformer architecture with learned global/local positional encoding, horizon-specific decoder-encoder attention, and decoder self-attention



(a) Decoder self-attention



(b) Forecasts fail to adjust as new information is observed



(c) Forecast is fluctuating even though observed demand is constant

Figure 4. The decoder attends over past forecasts for the *same target horizon* – the context h_t contains feedback information (demand and other encoded signals), allowing the model to adjust forecasts that are too volatile or not volatile enough as the target date approaches.

C. Large Scale Demand Forecasting Experiments

C.1. Ablation Studies

In the ablation studies, the model MQT-All uses a standard multi-head attention applied to the concatenated contexts for all horizons at each period t. A separate head is used for each output horizon (52 heads on the private dataset).

Table 5. Attention weight and output computations for MQT-All decoder self-attention

ATTENTION WEIGHTS	OUTPUT
$A^h_{t,s} = \mathbf{q}_t^\top \mathbf{W}_q^{h,\top} \mathbf{W}_k^h \mathbf{k}_s$ $\mathbf{q}_t = [\mathbf{c}_{s,1}; \cdots; \mathbf{c}_{s,H}; \mathbf{h}_t; \mathbf{r}_t]$ $\mathbf{k}_s = [\mathbf{c}_{s,1}; \cdots; \mathbf{c}_{s,H}; \mathbf{r}_s]$ $\mathbf{v}_s = [\mathbf{c}_{s,1}; \cdots; \mathbf{c}_{s,H}]$	$\widetilde{\mathbf{c}}_{t,h} = \sum_{s=t-L}^{t} A^h_{t,s} \mathbf{W}_v^h \mathbf{v}_s$

Table 6 gives the number of parameters in each trained model.

Table 6. Parameter counts in trained models for the large scale demand-forecasting task.

MODEL	NUMBER OF PARAMETERS
MQ-CNN	9.17×10^5
MQT	9.25×10^5
MQT-NODS	9.09×10^5
MQT-ALL	2.82×10^6

C.2. Experiment Setup

In this section we describe the details of the model architecture and training procedure used in the experiments on the large-scale demand forecasting application.

TRAINING PROCEDURE

Because we did not have enough history available to set aside a true holdout set, all models are trained for 100 epochs, and the final model is evaluated on the test set. For the same reason, no hyperparameter tuning was performed.

ARCHITECTURE AND HYPERPARAMETERS

The categorical variables consist of static features of the item, and the timeseries categorical variables are event indicators (e.g. holidays). The parameters are summarized in Table 7.

Table 7. Parameter settings for Large Scale Demand Forecasting Experiments

Parameter	Value
Encoder Convolution Dilation Rates	[1,2,4,8,16,32]
Position Encoding Dilation Rates	[1,2,4,8,16,20]
Static Categorical	One-Hot
Time-Series Categorical	One-Hot
Static Encoder Dimension	64
Convolution Filters	32
Attention Block Head Dimension	16
Dropout Rate	0.15
Activation Function	ReLU

C.3. Results for shorter horizons

Table 8 shows results along various dimensions for horizons up to 6 weeks. Similar to the results in Section 4, MQTransformer significantly outperforms the baseline.

Table 8. Quantile loss on the backtest year along different dimensions. Values are relative performance versus baseline, with the best result for each dimension emphasized.

	MQ-CNN		MQT		MQT-NoDS		MQT-All	
Dimension	P50	P90	P50	P90	P50	P90	P50	P90
Overall	1.000	1.000	**0.962**	**0.915**	0.983	0.963	0.977	0.957
Seasonal Peak 1	1.000	1.000	**0.865**	**0.667**	0.881	0.688	0.908	0.729
Seasonal Peak 2	1.000	1.000	**0.925**	**0.850**	0.972	0.910	0.932	0.886
Seasonal Peak 3	1.000	1.000	**0.798**	**0.804**	0.803	0.872	0.815	0.830
Peak Rampdown	1.000	1.000	0.968	**0.956**	0.983	0.996	**0.967**	0.975
Promotion Type 1	1.000	1.000	**0.915**	**0.601**	0.929	0.616	0.921	0.628
Promotion Type 2	1.000	1.000	**0.764**	**0.537**	0.805	0.561	0.861	0.640
Promotion Type 3	1.000	1.000	**0.901**	**0.746**	1.075	0.942	0.941	0.848

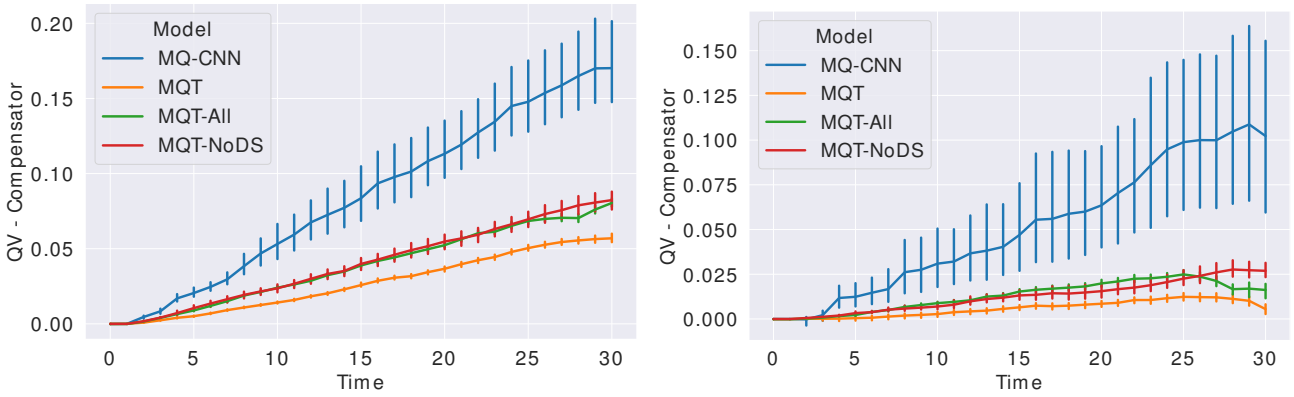


Figure 5. Martingale diagnostic process $\{V_t\}$ for P50 (left) and P90 (right) forecasts. Trajectories are demand-weighted and the results are averaged over all items, target weeks in the test period (2018-2019). Closer to zero is better.

C.4. Threshold Volatility Ablation Study

We study the effect of our proposed attention mechanisms on excess forecast volatility using diagnostic tools recently proposed by Foster and Stine (11). Coverage probabilities are computed by fitting a gamma distribution to the two quantiles produced for each FCT, target horizon. Figure 5 shows $\{V_t\}$ (see Appendix A.2) for the 4 models we consider, along with bootstrapped 95% confidence intervals. The results are weighted by demand – in the context of bull-whip effects in a supply chain, excess volatility will have the largest impact for items with high demand – but similar results are obtained if we weight all items equally. Under the MMFE, the lines should appear horizontal and any deviation above this (on an aggregate level) indicates excess volatility in the forecast evolution. While none of the models produce ideal forecasts, MQT has the least amount of excess volatility – it shows statistically significant reductions over all other models – and achieves a 67% reduction over baseline at P50 and 95% at P90.

D. Experiments on Public Datasets

In this section we describe the experiment setup used for the public datasets in Section 4. As mentioned in Appendix D.4, we display the baseline results published in Lim et al. (18). For MQ-CNN and MQTransformer, we use their pre-processing and evaluation code to ensure parity.

D.1. Datasets

We evaluate our MQTransformer on four public datasets. We summarize the datasets and preprocessing logic below; the reader is referred to Lim et al. (18) for more details. Lim et al. (18) released their code under the Apache 2.0 License. Favorita's terms of use for their retail dataset allows for it to be used for scientific research. The Electricity and Traffic datasets are provided by the UCI machine learning repository (10). The volatility dataset is sourced from the Oxford-Man Institute's realized library v0.3 (13).

RETAIL

This dataset is provided by the Favorita Corporacion (a major Grocery chain in Ecuador) as part of a Kaggle[1] to predict sales for thousands of items at multiple brick-and-mortar locations. In total there are 135K items (item, store combinations are treated as distinct entities), and the dataset contains a variety of features including: local, regional and national holidays; static features about each item; total sales volume at each location. The task is to predict log-sales for each (item, store) combination over the next 30 days, using the previous 90 days of history. The training period is January 1, 2015 through December 1, 2015. The following 30 days are used as a validation set, and the 30 days after that as the test set. These 30 day windows correspond to a single FCT. While Lim et al. (18) extract only 450K samples from the histories during the train window, there are in fact 20M trajectories avalaible for training – because our models can produce forecasts for multiple trajectories (FCDs) simultaneously, we train using all available data from the training window.

For the volatility analysis presented in Figure 6, we used a 60 day validation window (March 1, 2016 through May 1, 2016), which corresponds to 30 FCTs.

ELECTRICITY

This dataset consists of time series for 370 customers of at an hourly grain. The univariate data is augmented with a day-of-week, hour-of-day and offset from a fixed time point. The task is to predict hourly load over the next 24 hours for each customer, given the past seven days of usage. From the training period (January 1, 2014 through September, 1 2019) 500K samples are extracted.

TRAFFIC

This dataset consists of lane occupancy information for 440 San Francisco area freeways. The data is aggregated to an hourly grain, and the task is to predict the hourly occupancy over the next 24 hours given the past seven days. The training period consist of all data before 2008-06-15, with the final 7 days used as a validation set. The 7 days immediately following the training window is used for evaluation. The model takes as input lane occupancy, hour of day, day of week, hours from start and an entity identifier.

VOLATILITY

The volatility dataset consists of 5 minute sub-sampled realized volatility measurements from 2000-01-03 to 2019-06-28. Using the past one year's worth of daily measurements, the goal is to predict the next week's (5 business days) volatility. The period ending on 2015-12-31 is used as the training set, 2016-2017 as the validation set, and 2018-01-01 through 2019-06-28 as the evaluation set. The region identifier is provided as a static covariate, along with time-varying covariates daily returns, day-of-week, week-of-year and month. A log transformation is applied to the target.

D.2. Training Procedure

We only consider tuning two hyper-parameters, size of hidden layer $d_h \in \{32, 64, 128\}$ and learning rate $\alpha \in \{1 \times 10^{-2}, 1 \times 10^{-3}, 1 \times 10^{-4}\}$. The model is trained using the ADAM optimizer (16) with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e-8$ and a minibatch size of 256, for a maximum of 100 epochs and an early stopping patience of 5 epochs. We train a model for each hyperparameter setting in the search grid (6 combinations), select the one with the minimal validation loss and report the selected model's test-set error in Table 9.

[1]The original competition can be found here.

Table 9. Quantile loss metrics with the best results on each task emphasized. Results in parentheses correspond to training MQTransformer without forking sequences on 450K trajectories only.

	TASK	DEEPAR	CONVTRANS	MQ-RNN	MQ-CNN	TFT	MQTRANSFORMER
P50 QL	ELECTRICITY	0.075	0.059	0.077	0.076	**0.055**	0.057
	RETAIL	0.574	0.429	0.379	0.269	0.354	**0.256** (0.2645)
	VOLATILITY	0.050	0.047	0.042	0.042	**0.039**	**0.039**
	TRAFFIC	0.161	0.122	0.117	0.115	**0.095**	0.101
	TASK	DEEPAR	CONVTRANS	MQ-RNN	MQ-CNN	TFT	MQTRANSFORMER
P90 QL	ELECTRICITY	0.040	0.034	0.036	0.035	**0.027**	**0.027**
	RETAIL	0.230	0.192	0.152	0.118	0.147	**0.106** (0.109)
	VOLATILITY	0.024	0.024	0.021	0.020	0.020	**0.019**
	TRAFFIC	0.099	0.081	0.082	0.077	0.070	**0.068**

D.3. Architecture Details

Our MQTransformer architecture used for these experiments contain a single tune-able hyperparameter – hidden layer dimension d_h. Dataset specific settings are used for the dilation rates. For static categorical covariates we use an embedding layer with dimension d_h and use one-hot encoding for time-series covariates. A dropout rate of 0.15 and ReLU activations are used throughout the network. The only difference between this variant and the one used for the non-public large scale demand forecasting task is the use of an embedding layer for static, categorical covariates rather than one-hot encoding.

D.4. Results

Following Lim et al. (18), we consider applications to brick-and-mortar retail sales, electricity load, securities volatility and traffic forecasting. In each task, we report the quantile loss summed over all forecast creation times, normalized by the target values

$$\frac{2\sum_t \sum_k L_q\left(y_{t+k}, \widehat{y}^{(q)}_{t+k}\right)}{\sum_t \sum_k |y_{t+k}|}.$$

For the retail task, we predict the next 30 days of sales, given the previous 90 days of history. This dataset has a rich set of static, time series, and known features. At the other end of the spectrum, the electricity load dataset is univariate. See Appendix D for more information about these tasks. Table 9 compares MQTransformer's performance with DeepAR (24), ConvTrans (17), MQ-RNN (32), and TFT (18) [2].

Our MQTransformer architecture is competitive with or beats the state-of-the-art on the electricity load, volatility and traffic prediction tasks, as shown in Table 9. On the most challenging task, it dramatically outperforms the previously reported state of the art by 38% and the MQ-CNN baseline by 5% at P50 and 11% at P90. Because MQ-CNN and MQTransformer are trained using forking sequences, we can use the entire training population, rather than downsample as is required to train TFT (18). To ascertain what portion of the gain is due to learning from more trajectories, versus our innovations alone, we retrain the optimal MQTransformer architecture using a random sub-sample of 450K trajectories (the same sampling procedure as TFT) and without using forking sequences – the results are indicated in parentheses in Table 9. We can observe that MQTransformer still dramatically outperforms TFT, and its performance is similar to the MQ-CNN baseline trained on all trajectories. See Appendix D for more details and results on the Favorita forecasting task.

Computational Efficiency As an example take the Retail dataset. (18) was only able to use 450K out of 20M trajectories and the optimal TFT architecture required 13 minutes per epoch (minimum validation error at epoch 6)[3] using a single V100 GPU. Our innovations are compatible with forking sequences so our architecture can use *all available trajectories*. To train, MQTransformer requires only 5 minutes per epoch (on 20M trajectories) using a single V100 GPU (minimum validation error at 5 epochs).Some of the differences in runtime can be attributed to use of different deep learning frameworks, but it is clear that MQTransformer can be trained much more efficiently than models like TFT and DeepAR.

D.5. Reported Model Parameters

The optimal parameters for each task are given in Table 10.

Table 10. Parameter settings of reported MQTransformer model on each public dataset.

Name	d_h	α	Enc. Dilation Rates	Pos. Dilation Rates
Electricity	128	1×10^{-3}	[1,2,4,8,16,32]	[1,2,4,8,8]
Traffic	64	1×10^{-3}	[1,2,4,8,16,32]	[1,2,4,8,8]
Volatility	128	1×10^{-3}	[1,2,4,8,16,32,64]	[1,1,2]
Retail	64	1×10^{-4}	[1,2,4,8,16,32]	[1,2,4,8,14]

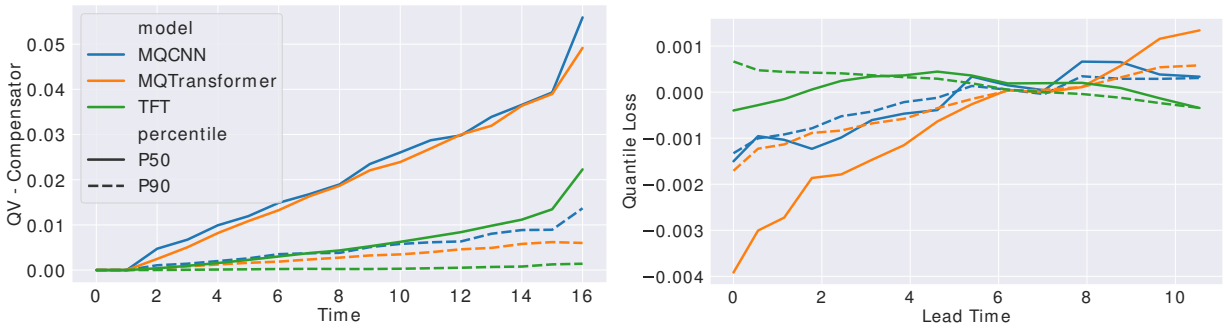


Figure 6. Forecast evolution analysis on the retail dataset. Left: Martingale Diagnostic Process $\{V_t\}$. Right: QL by lead time, averaged over target dates from 2016-03-01 through 2016-05-01; QL trajectories are centered around 0.

D.6. Martingale Diagnostics on Favorita Forecasting Task

On the Favorita retail forecasting task, Figure 6 shows that as expected, MQTransformer substantially reduces excess volatility in the forecast evolution compared to the MQ-CNN baseline. Somewhat surprisingly, TFT exhibits much lower volatility than does MQTransformer. In Figure 6, the right hand plot displays quantile loss as the target date approaches – trajectories for each model are zero centered to emphasize the trends exhibited. While TFT is less volatile, *it is also less accurate* as it fails to incorporate newly available information. By contrast, MQTransformer is both *less volatile* and *more accurate* when compared with MQ-CNN.

E. Non-Differentiable Cost Functions

In this section, we state and prove an alternative version of Theorem 3.1 for non-differentiable cost functions and discrete random variables X.

Theorem E.1. *Assume that $c : \mathbb{R} \to \mathbb{R}$ is a convex function, which is minimized at 0 and differentiable everywhere except possibly at 0. Assume that the subgradient of c at 0 is contained in the interval $[-L, U]$, i.e. $\partial c(0) \subseteq [-L, U]$, for constants $L, U \geq 0$. Further assume that for each $t = 1, \ldots, T$, the distribution of $X \mid \mathcal{F}_t$ is discrete and supported on the points $x_{1,t}, \ldots, x_{K_t,t} \in \mathbb{R}$, and $\Pr\{X = x_{k,t} \mid \mathcal{F}_t\} = p_{k,t}$ for $k = 1, \ldots, K_t$. Then*

$$\mathbb{E}[\mathrm{Vol}_{c_X}^{(t:T)}] = \mathbb{E}[c_X(\widehat{X}_t)] - \mathbb{E}[c_X(\widehat{X}_T)] + \mathrm{err}, \tag{12}$$

where

$$|\mathrm{err}| \leq (L + U) \sum_{t=1}^{T} \mathbb{E}\left[|\widehat{X}_{t-1} - \widehat{X}_t| p_{\widehat{k}_t,t} \right]$$

and $p_{\widehat{k}_t,t} = p_{k,t}$, if $\widehat{X}_t = x_{k,t}$ for some $k = 1, \ldots, K_t$, and 0 otherwise.

Proof. Since $\widehat{X}_t$ satisfies

$$\widehat{X}_t = \operatorname*{argmin}_{X_t \in \mathcal{F}_t} \mathbb{E}[c_X(X_t) \mid \mathcal{F}_t],$$

[2]Results for TFT, MQ-RNN, DeepAR and ConvTrans are from (18).
[3]Timing results obtained by running the source code provided by Lim et al. (18)

$0 \in \partial \mathbb{E}[c_X(\widehat{X}_t) \mid \mathcal{F}_t]$. Additionally,

$$\partial \mathbb{E}[c_X(\widehat{X}_t) \mid \mathcal{F}_t] = \begin{cases} \{0\} & \text{if } \widehat{X}_t \notin \{x_{1,t}, \ldots, x_{K_t,t}\} \\ \sum_{k' \neq k} p_{k',t} c'(\widehat{X}_t - x_{k',t}) + p_{k,t}[-L, U] & \text{if } \widehat{X}_t = x_{k,t}, \end{cases}$$

and $\mathbb{E}[c'_X(\widehat{X}_t) \mid \mathcal{F}_t] \in \partial \mathbb{E}[c_X(\widehat{X}_t) \mid \mathcal{F}_t]$. It follows that $|\mathbb{E}[c'_X(\widehat{X}_t) \mid \mathcal{F}_t]| \leq p_{\widehat{k}_t,t}(L+U)$. Thus,

$$\begin{aligned} \left|\mathbb{E}[c'_X(\widehat{X}_t)(\widehat{X}_{t-1} - \widehat{X}_t) \mid \mathcal{F}_t]\right| &= \left|(\widehat{X}_{t-1} - \widehat{X}_t)\mathbb{E}[c'_X(\widehat{X}_t) \mid \mathcal{F}_t]\right| \\ &\leq |\widehat{X}_{t-1} - \widehat{X}_t|(L+U)p_{\widehat{k}_t,t}. \end{aligned}$$

Plugging this in to

$$\mathbb{E}[c_X(\widehat{X}_t)] = \mathbb{E}[c_X(\widehat{X}_T)] + \sum_{t'=t}^{T-1} \mathbb{E}[c'_X(\widehat{X}_{t'+1})(\widehat{X}_{t'} - \widehat{X}_{t'+1})] + \mathbb{E}[\mathrm{Vol}_{c_X}^{(t:T)}],$$

we obtain

$$\mathbb{E}[\mathrm{Vol}_{c_X}^{(t:T)}] = \mathbb{E}[c_X(\widehat{X}_t)] - \mathbb{E}[c_X(\widehat{X}_T)] + \mathrm{err},$$

where

$$|\mathrm{err}| \leq (L+U) \sum_{t=1}^{T} \mathbb{E}\left[|\widehat{X}_{t-1} - \widehat{X}_t| p_{\widehat{k}_t,t}\right].$$

This proves the theorem. □

Next we work out how to apply Theorem E.1 for quantile loss and the Forecast Regression methodology, in the case where replicate observations and forecasts are available. Assume that c is quantile loss, i.e. $c(x) = (1-q)xI\{x > 0\} - qxI\{x < 0\}$ for some fixed $0 \leq q \leq 1$. Then c is non-differentiable at 0 and $\partial c(0) = [-q, 1-q]$. Assume further that we have replicates $X^{(i)}$, $i = 1, \ldots, N$ and corresponding forecasts $\widehat{X}_t^{(i)} \in \mathcal{F}_t$, $t = 1, \ldots, T$. As in Section 3.1, we consider the alternative forecasts $\widetilde{X}_t^{(i)} = \sum_{s=1}^{t} \beta_{t',t} \widehat{X}_{t'}^{(i)}$, where $\boldsymbol{b}_t = (\beta_{1,t}, \ldots, \beta_{t,t})$ satisfies

$$\boldsymbol{b}_t \in \operatorname*{argmin}_{\boldsymbol{b} \in \mathbb{R}^t} \frac{1}{N} \sum_{i=1}^{N} c_{X^{(i)}} \left(\sum_{t'=1}^{t} \beta_{t',t} \widehat{X}_{t'}^{(i)} \right).$$

Then, by (12),

$$\frac{1}{N} \sum_{i=1}^{N} \left\{ \mathrm{Vol}_{c_{X^{(i)}}}^{(1:T)}(\widetilde{X}_1^{(i)}, \ldots, \widetilde{X}_T^{(i)}) - [c_{X^{(i)}}(\widetilde{X}_1^{(i)}) - c_{X^{(i)}}(\widetilde{X}_T^{(i)})] \right\} = \mathrm{err},$$

where

$$|\mathrm{err}| \leq \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T-1} |\widetilde{X}_t^{(i)} - \widetilde{X}_{t+1}^{(i)}| I\{\widetilde{X}_{t+1}^{(i)} = X^{(i)}\}.$$

F. Quantile Loss Bregman Divergence Calculation

We provide the calculation for the Bregman divergence for the quantile loss.

$$D_{c_X}(\widehat{X}_t, \widehat{X}_{t+1}) = \begin{cases} |\widehat{X}_t - X| & \text{if } \mathrm{sgn}(\widehat{X}_t - X) = -\mathrm{sgn}(\widehat{X}_{t+1} - X), \\ 0 & \text{if } \mathrm{sgn}(\widehat{X}_t - X) = \mathrm{sgn}(\widehat{X}_{t+1} - X), \\ c_X(\widehat{X}_t) & \text{if } \mathrm{sgn}(\widehat{X}_t - X)\mathrm{sgn}(\widehat{X}_{t+1} - X) = 0. \end{cases} \tag{13}$$

Let $c(x) = (1-q)xI\{x > 0\} - qxI\{x < 0\}$ be the q-th quantile loss function, where $0 \leq q \leq 1$ is fixed. The Bregman

divergence associated with $c_X(x) = c(x - X)$ is

$$\begin{aligned}
D_{c_X}(\widehat{X}_t, \widehat{X}_{t+1}) &= q(X - \widehat{X}_t)I\{\widehat{X}_t < X\} + (1-q)(\widehat{X}_t - X)I\{X < \widehat{X}_t\} \\
&\quad - q(X - \widehat{X}_{t+1})I\{\widehat{X}_{t+1} < X\} - (1-q)(\widehat{X}_{t+1} - X)I\{X < \widehat{X}_{t+1}\} \\
&\quad - \left[-qI\{\widehat{X}_{t+1} < X\} + (1-q)I\{X < \widehat{X}_{t+1}\}\right](\widehat{X}_t - \widehat{X}_{t+1}) \\
&= q(X - \widehat{X}_t)I\{\widehat{X}_t < X\} + (1-q)(\widehat{X}_t - X)I\{X < \widehat{X}_t\} \\
&\quad - q(X - \widehat{X}_{t+1})I\{\widehat{X}_{t+1} < X\} - (1-q)(\widehat{X}_{t+1} - X)I\{X < \widehat{X}_{t+1}\} \\
&\quad - \left[-qI\{\widehat{X}_{t+1} < X\} + (1-q)I\{X < \widehat{X}_{t+1}\}\right](\widehat{X}_t - X + X - \widehat{X}_{t+1}) \\
&= q(X - \widehat{X}_t)I\{\widehat{X}_t < X\} + (1-q)(\widehat{X}_t - X)I\{X < \widehat{X}_t\} \\
&\quad + qI\{\widehat{X}_{t+1} < X\}(\widehat{X}_t - X) - (1-q)I\{X < \widehat{X}_{t+1}\}(\widehat{X}_t - X) \\
&= q(X - \widehat{X}_t)[I\{\widehat{X}_t < X\} - I\{\widehat{X}_{t+1} < X\}] \\
&\quad + (1-q)(\widehat{X}_t - X)[I\{X < \widehat{X}_t\} - I\{X < \widehat{X}_{t+1}\}] \\
&= \begin{cases} |\widehat{X}_t - X| & \text{if } \operatorname{sgn}(\widehat{X}_t - X) = -\operatorname{sgn}(\widehat{X}_{t+1} - X), \\ 0 & \text{if } \operatorname{sgn}(\widehat{X}_t - X) = \operatorname{sgn}(\widehat{X}_{t+1} - X), \\ c_X(\widehat{X}_t) & \text{if } \operatorname{sgn}(\widehat{X}_t - X)\operatorname{sgn}(\widehat{X}_{t+1} - X) = 0. \end{cases}
\end{aligned}$$

Now let F_t denote the cumulative distribution function of $X \mid \mathcal{F}_t$. The expected Bregman divergence is

$$\begin{aligned}
\mathbb{E}[D_{c_X}(\widehat{X}_t, \widehat{X}_{t+1}) \mid \mathcal{F}_{t+1}] &= \int |\widehat{X}_t - x| I\{\operatorname{sgn}(\widehat{X}_t - x) = -\operatorname{sgn}(\widehat{X}_{t+1} - x)\}\, dF_{t+1}(x) \\
&\quad + \int c_x(\widehat{X}_t) I\{\operatorname{sgn}(\widehat{X}_t - x)\operatorname{sgn}(\widehat{X}_{t+1} - x) = 0\}\, dF_{t+1}(x) \\
&= \int (x - \widehat{X}_t) I\{\widehat{X}_t < x \le \widehat{X}_{t+1}\}\, dF_{t+1}(x) \\
&\quad + \int (\widehat{X}_t - x) I\{\widehat{X}_{t+1} < x \le \widehat{X}_t\}\, dF_{t+1}(x) \\
&\quad + c(\widehat{X}_t - \widehat{X}_{t+1}) \Pr\{X = \widehat{X}_{t+1} \mid \mathcal{F}_{t+1}\} \\
&\quad - (\widehat{X}_{t+1} - \widehat{X}_t) \Pr\{X = \widehat{X}_{t+1} \mid \mathcal{F}_{t+1}\} I\{\widehat{X}_t < \widehat{X}_{t+1}\} \\
&= \int \int I\{\widehat{X}_t < u < x\} I\{\widehat{X}_t < x \le \widehat{X}_{t+1}\}\, du\, dF_{t+1}(x) \\
&\quad + \int \int I\{x \le u < \widehat{X}_t\} I\{\widehat{X}_{t+1} < x \le \widehat{X}_t\}\, du\, dF_{t+1}(x) \\
&\quad + (1-q)(\widehat{X}_t - \widehat{X}_{t+1}) \Pr\{X = \widehat{X}_{t+1} \mid \mathcal{F}_{t+1}\} \\
&= \int I\{\widehat{X}_t < u < \widehat{X}_{t+1}\}[F_{t+1}(\widehat{X}_{t+1}) - F_{t+1}(u)]\, du \\
&\quad + \int I\{\widehat{X}_{t+1} < u < \widehat{X}_t\}[F_{t+1}(u) - F_{t+1}(\widehat{X}_{t+1})]\, du \\
&\quad + (1-q)(\widehat{X}_t - \widehat{X}_{t+1}) \Pr\{X = \widehat{X}_{t+1} \mid \mathcal{F}_{t+1}\} \\
&= \int_{\widehat{X}_{t+1}}^{\widehat{X}_t} F_{t+1}(u) - F_{t+1}(\widehat{X}_{t+1})\, du \\
&\quad + (1-q)(\widehat{X}_t - \widehat{X}_{t+1}) \Pr\{X - \widehat{X}_{t+1} \mid \mathcal{F}_{t+1}\}
\end{aligned}$$

This calculation shows that we can still compute $\mathbb{E}[D_{c_X}(\widehat{X}_t, \widehat{X}_{t+1}) \mid \mathcal{F}_{t+1}]$ even though for quantile loss the Bregman volatility depends on the final demand.

Note that the quantity on the right-hand side, the expected Bregman divergence for quantile loss is similar to the quantile divergence defined in (23) and studied in (31) and elsewhere.