MQTRANSFORMER: MULTI-HORIZON FORECASTS WITH CONTEXT DEPENDENT AND FEEDBACK-AWARE ATTENTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advances in neural forecasting have produced major improvements in accuracy for probabilistic demand prediction. In this work, we propose novel improvements to the current state of the art by incorporating changes inspired by recent advances in Transformer architectures for Natural Language Processing. We develop a novel decoder-encoder attention for context-alignment, improving forecasting accuracy by allowing the network to study its own history based on the context for which it is producing a forecast. We also present a novel positional encoding that allows the neural network to learn context-dependent seasonality functions as well as arbitrary holiday distances. Finally we show that the current state of the art MQ-Forecaster (Wen et al., 2017) models display excess variability by failing to leverage previous errors in the forecast to improve accuracy. We propose a novel decoder-self attention scheme for forecasting that produces significant improvements in the excess variation of the forecast.

1 INTRODUCTION

Time series forecasting is a fundamental problem in machine learning with relevance to many application domains including supply chain management, finance, healthcare analytics, and more. Modern forecasting applications require predictions of many correlated time series over multiple horizons. In multi-horizon forecasting, the learning objective is to produce forecasts for multiple future horizons at each time-step. Beyond simple point estimation, decision making problems require a measure of uncertainty about the forecasted quantity. Access to the full distribution is usually unnecessary, and several quantiles are sufficient (many problems in Operations Research use the 50th and 90th percentiles, for example).

As a motivating 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 to maximize revenue and/or customer value. Accurate forecasts are important, but – perhaps less obviously – forecasts that don't exhibit excess volatility as a target date approaches minimize costly, bull-whip effects in a supply chain (Chen et al., 2000; Bray and Mendelson, 2012).

Recent work applying deep learning to time-series forecasting focuses primarily on the use of recurrent and convolutional architectures (Nascimento et al., 2019; Yu et al., 2017; Gasparin et al., 2019; Mukhoty et al., 2019; Wen et al., 2017)[1]. These are Seq2Seq architectures (Sutskever et al., 2014) – which consist of an *encoder* which takes an input sequence and summarizes it into a fixed-length context vector, and a *decoder* which produces an output sequence. It is well known that Seq2Seq models suffer from an information bottleneck by transmitting information from encoder to decoder via a single hidden state. To address this Bahdanau et al. (2014) introduces a method called *attention*, allowing the decoder to take as input a weighted combination of relevant latent encoder states at each output time step, rather than using a single context to produce all decoder outputs. While NLP is the predominate application of attention architectures, in this paper we show how novel attention modules and positional embeddings can be used to introduce proper inductive biases for probabilistic time-series forecasting to the model architecture.

[1]For a complete overview see Benidis et al. (2020)

Even with these shortcomings, this line of work has lead to major advances in forecast accuracy for complex problems, and real-world forecasting systems increasingly rely on neural nets. Accordingly, a need for black-box forecasting system diagnostics has arisen. Stine and Foster (2020b;a) use probabilistic martingales to study the dynamics of forecasts produced by an arbitrary forecasting system. They can be used to detect the degree to which forecasts adhere to the martingale model of forecast evolution (Heath and Jackson, 1994) and to detect unnecessary volatility (above and beyond any inherent uncertainty) in the forecasts produced. Thus, Stine and Foster (2020b;a) describe a way to connect the excess variation of a forecast to accuracy misses against the realized target.

While, multi-horizon forecasting networks such as (Wen et al., 2017; Madeka et al., 2018), minimize quantile loss - the network architectures do not explicitly handle excess variation, since forecasts on any particular date are not made aware of errors in the forecast for previous dates. In short, such tools can be used to detect flaws in forecasts, but the question of how to incorporate that information into model design is unexplored.

Our 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 (Wen et al., 2017). The major contributions of this paper are

1. **Positional Encoding from Event Indicators**: Current MQ-Forecasters use explicitly engineered holiday "distances" to provide the model with information about the seasonality of the time series. We introduce a novel positional encoding mechanism that allows the network to learn a seasonality function depending on other information of the time series being forecasted, and demonstrate that its a strict generalization of conventional position encoding schemes.
2. **Horizon-Specific Decoder-Encoder Attention**: Wen et al. (2017); Madeka et al. (2018) and other MQ-Forecasters learn a single encoder representation for all future dates and periods being forecasted. We present a novel horizon-specific decoder-encoder attention scheme that allows the network to learn a representation of the past that depends on which period is being forecasted.
3. **Decoder Self-Attention for Forecast Evolution**: To the best of our knowledge, this is *the first work to consider the impacts of network architecture design on forecast evolution*. Importantly, we accomplish this by using attention mechanisms to introduce the right inductive biases, and not by explicitly penalizing a measure of forecast variability. This allows us to maintain a single objective function without needing to make trade-offs between accuracy and volatility.

By providing MQ-Forecasters with the structure necessary to learn *context information dependent encodings*, we observe major increases in accuracy (3.9% in overall P90 quantile loss throughout the year, and up to 60% during peak periods) on our demand forecasting application along with a significant reduction in excess volatility (52% reduction in excess volatility at P50 and 30% at P90). We also apply MQTransformer to four public datasets, and show parity with the state-of-the-art on simple, univariate tasks. On a substantially more complex public dataset (retail forecasting) we demonstrate a **38% improvement** over the previously reported state-of-the-art, and a 5% improvement in P50 QL, 11% in P90 QL versus our baseline. Because our innovations are compatible with efficient training schemes, our architecture also achieves a significant speedup (several orders of magnitude greater throughput) over earlier transformer models for time-series forecasting.

2 Background and Related Work

2.1 Time Series Forecasting

Formally, the task considered in our work is the high-dimensional regression problem

$$p(y_{t+1,i}, \ldots, y_{t+H,i} | \mathbf{y}_{:t,i}, \mathbf{x}^{(h)}_{:t,i}, \mathbf{x}^{(f)}_{:t,i}, \mathbf{x}^{(s)}_{i}), \tag{1}$$

where $y_{t+s,i}$, $\mathbf{y}_{:t,i}$, $\mathbf{x}^{(h)}_{:t,i}$, $\mathbf{x}^{(f)}_{t:,i}$, $\mathbf{x}^{(s)}$ denote future observations of the target time series i, observations of the target time series observed up until time t, the past covariates, known future information, and static covariates, respectively.

For sequence modeling problems, Seq2Seq (Sutskever et al., 2014) is the canonical deep learning framework and although applied this architecture to neural machine translation (NMT) tasks, it has

since been adapted to time series forecasting (Nascimento et al., 2019; Yu et al., 2017; Gasparin et al., 2019; Mukhoty et al., 2019; Wen et al., 2017; Salinas et al., 2020; Wen and Torkkola, 2019). The MQ-Forecaster framework (Wen et al., 2017) solves (1) above by treating each series i as a sample from a joint stochastic process and feeding into a neural network which predicts Q quantiles for each horizon. These types of models, however, inherit from the Seq2Seq architecture the limited contextual information available to the decoder as it produces each estimate $y^q_{t+s,i}$, the qth quantile of the distribution of the target at time $t+s$ $y_{t+s,i}$. Seq2Seq models rely on a single encoded context to produce forecasts for all horizons, imposing an information bottleneck and making it difficult for the model to understand long term dependencies.

Our MQTransformer architecture, like other MQ-Forecasters, uses the direct strategy: the model outputs the quantiles of interest directly, rather than the parameters of a distribution from which samples are to be generated. This has been shown (Wen et al., 2017) to outperform parametric models, like DeepAR (Salinas et al., 2020), on a wide variety of tasks. Recently, Lim et al. (2019) consider an application of attention to multi-horizon forecasting, but their method still produces a single context for all horizons. Furthermore, by using an RNN decoder their models do not enjoy the same scaling properties as MQ-Forecaster models. To the best of our knowledge, our work is the first to devise attention mechanisms for this problem that readily scale.

2.2 Attention Mechanisms

Bahdanau et al. (2014) introduced the concept of an attention mechanism to solve the information bottleneck and sequence alignment problems in Seq2Seq architectures for NMT. Recently, attention has enjoyed success across a diverse range of applications including natural language processing (NLP), computer vision (CV) and time-series forecasting tasks (Galassi et al., 2019; Xu et al., 2015; Shun-Yao Shih and Fan-Keng Sun and Hung-yi Lee, 2019; Kim and Kang, 2019; Cinar et al., 2017; Li et al., 2019; Lim et al., 2019). Many variants have been proposed including self-attention and dot-product attention (Luong et al., 2015; Cheng et al., 2016; Vaswani et al., 2017; Devlin et al., 2019), and transformer architectures (end-to-end attention with no recurrent layers) achieve state-of-the-art performance on most NLP tasks.

Time series forecasting applications exhibit seasonal trends and the absolute position encodings commonly used in the literature cannot be applied. Our work differs from previous work on *relative position encodings* (Dai et al., 2019; Huang et al., 2018; Shaw et al., 2018) in that we learn a representation from a time series of indicator variables which encode events relevant to the target application (such as holidays and promotions). If event indicators relevant to the application are provided, then this imposes a strong inductive bias that will allow the model to generalize well to future observations. Existing encoding schemes either involve feature engineering (e.g. sinusoidal encodings) or have a maximum input sequence length, ours requires no feature engineering – the model learns it directly from raw data – and it extends to arbitrarily long sequences.

In the vanilla transformer (Vaswani et al., 2017), 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

$$A^h_{s,t} = (\mathbf{x}_s + \mathbf{r}_s)^\top \mathbf{W}^{h,\top}_q \mathbf{W}^h_k (\mathbf{x}_t + \mathbf{r}_t) \tag{2}$$

where $\mathbf{x}_s$, $\mathbf{r}_s$ are the observation of the time series and the position encoding, respectively, at time s. Section 3 introduces attention mechanisms that differ in their treatment of the position dependent biases. See Appendix A for additional discussion of attention mechanisms.

2.3 Martingale Diagnostics

Originally the *martingale model of forecast evolution* (MMFE) was conceived as a way to simulate demand forecasts used in inventory planning problems (Heath and Jackson, 1994). 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 (1991) for a thorough coverage in discrete time.

Taleb (2018) describe how martingale forecasts correspond to rational updating, then expanded by Augenblick and Rabin (2019). Taleb (2018), Taleb and Madeka (2019) and Augenblick and Rabin (2019) go on to develop tests for forecasts that rule out martingality and indicate irrational or predictable updating for binary bets. Stine and Foster (2020a;b) further extend these ideas to quantile forecasts. Specifically, 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, $\mathbb{E}[Q_T] = \pi(1-\pi)$.

A second quantity of interest is the martingale $V_t := Q_t - (p_t - \pi)^2$ which follows the typical structure of subtracting the compensator to turn a sub-martingale into a martingale. In Section 4 we leverage the properties of $\{V_t\}$ and $\{Q_t\}$ 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.

3 Methodology

As mentioned, this work is motivated in part by the needs of the consumers of forecasting systems. We therefore care about whether or not our innovations can be used in practice. Our methodology must scale to forecasting tens of thousands or millions of signals, at hundreds of horizons. We extend the MQ-Forecaster family of models (Wen et al., 2017) because it, unlike many other architectures considered in the literature, can be applied at a large-scale (millions of samples) due to its use of *forking sequences* – a technique to dramatically increase the effective batch size during training and avoid expensive data augmentation. In this section we present our MQTransformer architecture, building upon the MQ-Forecaster framework.

For ease of exposition, we reformulate the generic probabilistic forecasting problem in (1) as $p(y_{t+1,i}, \ldots, y_{t+H,i} | \mathbf{y}_{:t,i}, \mathbf{x}_{:t,i}, \mathbf{x}_i^{(l)}, \mathbf{x}^{(g)}, \mathbf{s}_i)$, 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. In this setting, known signifies that the model has access to (potentially noisy) observations of past and future values. Note that this formulation is equivalent to (1), and that 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.

3.1 Learning Objective

We train a quantile regression model to minimize the quantile loss, summed over all forecast creation dates and quantiles $\sum_t \sum_q \sum_k L_q(y_{t+k}, \widehat{y}_{t+k}^{(q)})$, where $L_q(y, \widehat{y}) = q(y - \widehat{y})_+ + (1-q)(\widehat{y} - y)_+$, $(\cdot)_+$ is the positive part operator, q denotes a quantile, and k denotes the horizon.

3.2 Network Architecture

The design of the architecture is similar to MQ-RNN (Wen et al., 2017), and consists of encoder, decoder and position encoding blocks (see Figure 4 in Appendix B). The position encoding outputs, for each time step t, are 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). In both cases, 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.

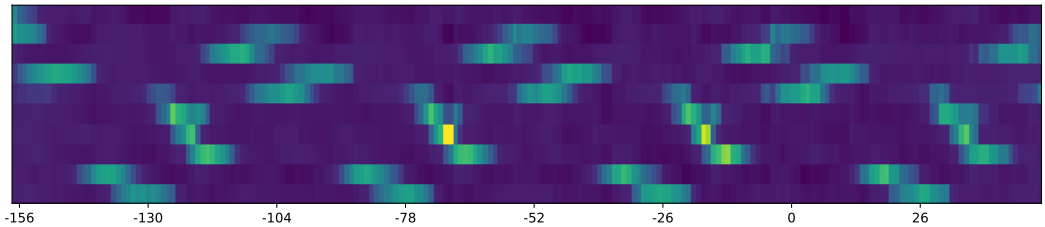

Figure 1: Position encoding learned from daily-grain event indicators

In this section we focus on our novel attention blocks and position encodings; the reader is directed to Appendix B for the other architecture details.

MQTransformer Now we describe a design, evaluated in Section 4, following the generic pattern given above. We 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 (van den Oord et al., 2016; Wen et al., 2017) to encode historical time-series and a multi-layer perceptron to encode the static features as (3).

Table 1: MQTransformer encoder and decoder

ENCODER		DECODER CONTEXTS	
$\mathbf{h}_t^1 = \text{TEMPORALCONV}(\mathbf{y}_{:t}, \mathbf{x}_{:t}, \mathbf{r}_{:t})$	(3)	$\mathbf{c}_{t,h}^{hs} = \text{HSATTENTION}(\mathbf{h}_{:t}, \mathbf{r})$	(4)
$\mathbf{h}_t^2 = \text{FEEDFORWARD}(\mathbf{s})$		$\mathbf{c}_t^a = \text{FEEDFORWARD}(\mathbf{h}_t, \mathbf{r})$	
$\mathbf{h}_t = [\mathbf{h}_t^1; \mathbf{h}_t^2]$,		$\mathbf{c}_t = [\mathbf{c}_{t,1}^a; \cdots; \mathbf{c}_{t,H}^{hs}; \mathbf{c}_t^a]$	
		$\widetilde{\mathbf{c}}_{t,h} = \text{DSATTENTION}(\mathbf{c}_{:t}, \mathbf{h}_{:t}, \mathbf{r})$,	

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 ($\mathbf{c}_{t,h}^{hs}$) and horizon agnostic ($\mathbf{c}_t^a$) contexts. Formally, the global branch $\mathbf{c}_t := m_G(\cdot)$ is given by (4).

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 $(\widehat{y}_{t+h}^1, \ldots, \widehat{y}_{t+h}^Q) = m_L(\mathbf{c}_t^a, \mathbf{c}_{t,h}^{hs}, \widetilde{\mathbf{c}}_{t,h}, \mathbf{r}_{t+h})$, where $\mathbf{c}_{:t}$ denotes the output of the global branch, up through the FCT t. Next we describe the specifics of our position encoding and attention blocks.

3.3 LEARNING POSITION AND CONTEXT REPRESENTATIONS FROM EVENT INDICATORS

Prior work typically uses a variant on one of two approaches to provide attention blocks with position information: (1) a handcrafted representation (such as sinusoidal encodings) or (2) a matrix $\mathbf{M} \in \mathbb{R}^{L \times d}$ of position encoding where L is the maximum sequence length and each row corresponds to the position encoding for time point.

In contrast, our novel encoding scheme maps sequences of indicator variables to a d-dimensional representations. For demand forecasting, this enables our model to learn an arbitrary function of events (like holidays and promotions) to encode position information. As noted above, our model includes two position encodings: $r_t^{(g)} := PE_t^{(g)}(\mathbf{x}^{(g)})$ and $r_t^{(l)} := PE_t^{(l)}(\mathbf{x}^{(l)})$, one that is shared among all time-series i and one that is specific. For the design we use in Section 4, $PE^{(g)}$ is implemented as a bidirectional 1-D convolution (looking both forward and backward in time) and $PE^{(l)}$ is an MLP applied separately at each time step. Figure 1 shows an example of $PE^{(g)}$ learned from holiday indicator variables. For reference, MQ-(C)RNN (Wen et al., 2017) uses linear holiday and promotion distances to represent position information.

Connection to matrix embeddings Another way to view our position encoding scheme is as a form of set based indexing into rows of an infinite dimensional matrix. We note that the traditional method of learning a matrix embedding $\mathbf{M}$ can be recovered as a special case of our approach. Consider a sequence of length L, and take $\mathbf{x}^{(g)} := [\mathbf{e}_1, \ldots, \mathbf{e}_L]$ where $\mathbf{e}_s$ is used to denote the vector in $\mathbb{R}^L$ with a 1 in the sth position and 0s elsewhere. To recover the matrix embedding scheme, we

define $\text{PE}_t^{\text{matrix}}(\mathbf{x}^{(g)}) := \mathbf{x}_t^{(g),\top}\mathbf{M}$. Thus we see that our scheme is a strict generalization of the matrix embedding approach commonly used in the NLP community.

3.4 Context Dependent and Feedback-Aware Attention

Table 2: Attention weight and output computations for blocks introduced in Section 3.4

Block	Attention Weights		Output	
Decoder-Encoder Attention	$A_{t,s}^h = \mathbf{q}_t^{h,\top}\mathbf{W}_q^\top\mathbf{W}_k\mathbf{k}_s$ $\mathbf{q}_t^h = [\mathbf{h}_t; \mathbf{r}_t; \mathbf{r}_{t+h}]$ $\mathbf{k}_s = [\mathbf{h}_s; \mathbf{r}_s]$ $\mathbf{v}_s = \mathbf{h}_s$	(5)	$\mathbf{c}_{t,h}^{hs} = \sum_{s=t-L}^{t} A_{t,s}^h \mathbf{v}_s$	(6)
Decoder Self-Attention	$A_{t,s,r}^h = \mathbf{q}_{t,h}^\top\mathbf{W}_q^{h,\top}\mathbf{W}_k^h\mathbf{k}_{s,r}$ $\mathbf{q}_{t,h} = [\mathbf{h}_t; \mathbf{c}_{t,h}^{hs}; \mathbf{r}_t; \mathbf{r}_{t+h}]$ $\mathbf{k}_{s,r} = [\mathbf{c}_{s,r}^{hs}; \mathbf{r}_s; \mathbf{r}_{s+r}]$ $\mathbf{v}_{s,r} = \mathbf{c}_{s,r}^{hs}$	(7)	$\tilde{\mathbf{c}}_{t,h}^{hs} = \sum_{(s,r)\in\mathcal{H}(t,h)} A_{s,t,r}^h \mathbf{v}_{s,r},$ $\mathcal{H}(t,h) := \{(s,r) \vert s+r=t+h\}$	(8)

Horizon-Specific Decoder-Encoder Attention Our horizon-specific attention mechanism is a multi-headed attention mechanism where the projection weights are shared across all horizons. Each head corresponds to a different horizon. It differs from a traditional multi-headed attention mechanism in that its purpose is to attend over representations of past time points to produce a representation specific to the target period. In our architecture, the inputs to the block are the encoder hidden states and position encodings. Mathematically, for time s and horizon h, the attention weight for the value at time t is computed as (5).

Observe that there are two key differences between these attention scores and those in the vanilla transformer architecture: (a) projection weights are shared by all H heads, (b) the addition of the position encoding of the target horizon h to the query. The output of our horizon specific decoder-encoder attention block, $\mathbf{c}_{t,h}^{hs}$, is obtained by taking a weighted sum of the encoder hidden contexts, up to a maximum look-back of L periods as in (6).

Decoder Self-Attention The martingale diagnostic tools developed in (Stine and Foster, 2020b) indicate 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. This is because the first model fails to learn from its past forecasts - it continues jumping between 40, 60 when the demand is 50 units.

In order to ameliorate this, we need to pass the information of the previous forecast errors into the current forecast. For each FCT t and horizon h, the model attends on the previous forecasts using a query containing the demand information for that period. The attention mechanism has a separate head for each horizon.

Rather than attend on the demand information and prior outputs directly, a richer representations of the same information is used: the demand information at time t is incorporated via the encoded context $\mathbf{h}_t$ and previous forecasts are represented via the corresponding horizon-specific context $\mathbf{c}_{s,r}^{hs}$ – in the absence of decoder-self attention $\mathbf{c}_{s,r}^{hs}$ would be passed through the local MLP to generate the forecasts. Formally, the attention scores are given by (7). The horizon-specific and feedback-aware outputs, $\tilde{\mathbf{c}}_{t,h}^{hs}$, are given by (8). Note how we sum only over previous forecasts of the same period.

Table 3: Aggregate Quantile Loss Metrics

MODEL	ALL LTSP	LTSP 0/4	SEASONAL PEAK 1	POST-PEAK RAMPDOWN	PROMOTION TYPE 1
BASELINE	1.000	1.000	1.000	1.000	1.000
DEC-ENC	**0.984**	0.931	0.748	0.712	0.706
DEC-ENC + DEC-SELF	0.989	**0.908**	**0.698**	**0.639**	**0.670**

4 EMPIRICAL RESULTS

4.1 LARGE-SCALE DEMAND FORECASTING

First, we evaluate our architecture on a demand forecasting problem for a large-scale e-commerce retailer with the objective of producing multi-horizon forecasts that span up to one year. Each horizon is specified by a lead time (LT), number of periods from the FCT to the start of the horizon, and a span (SP), number of periods covered by the forecast, combination. To assess the effects of each innovation, we ablate by removing components one at a time. MQTransformer is denoted as `Dec-Enc & Dec-Self Att`, `Dec-Enc Att` – which contains only the horizon-specific decoder-encoder unit – and `Baseline` – the vanilla MQ-CNN model. MQ-CNN is selected as the baseline since prior work[2] demonstrate that MQ-CNN outperforms MQ-RNN and DeepAR on this dataset, and as can be seen in Table 4, MQ-CNN similarly outperforms MQ-RNN and DeepAR on public datasets.

We conduct our experiments on a subset of products ($\sim$ 2 million products) in the US store. Each model is trained using a single machine with 8 NVIDIA V100 Tensor Core GPUs, on three years of demand data (2015-2018); one year (2018-2019) is held out for back-testing.

Forecast Accuracy Table 3 summarizes several key metrics that demonstrate the accuracy improvements achieved by adding our proposed attention mechanisms to the MQ-CNN architecture – the full set of results can be found in Appendix C. Introducing the Horizon-Specific Decoder-Encoder attention alone yields improvements along all metrics evaluated. Overall we see a 1.6% improvement in P50 QL and a 3.9% improvement in P90 QL. Notably, the attention mechanism yields significant improvements on short LT-SP (LT-SP 0/4).

Further, Table 3 demonstrates improved performance on seasonal peaks and promotions. Observe that while MQ-CNN performs well on some seasonal peaks, it also is misaligned and fails to ramp-down post-peak – post-peak ramp-down issues occur when the model continues to forecast high for target weeks after the peak week. By including MQTransformer's attention mechanisms in the architecture, we see a 43% improvement for Seasonal Peak 1 and a 56% improvement on Post-Peak Rampdown. In retail, promotions are used to provide a demand lift for products. Accordingly, a model should be able to react to the upcoming promotion and forecast an accurate lift in demand for the target weeks in which the promotion is placed. From Table 3 we see that MQTransformer achieves a see a 49% on items with Promotion Type 1 versus the baseline.

Forecast Volatility We study the effect of our proposed attention mechanisms on excess forecast volatility using diagnostic tools recently proposed by Stine and Foster (2020b;a). Figure 2 plots the process $\{V_t\}$ (see Section 2). In the plot, the lines should appear horizontal under the MMFE. Any deviation above this (on an aggregate level) indicates excess volatility in the forecast evolution. We can observe that while none of the models produce ideal forecasts, both attention models outperform the Baseline with the attention model with both proposed attention mechanisms performing the best in terms of these evolution diagnostics.

The green line corresponds to the attention model with only the horizon-specific decoder-encoder attention. We can see that compared to the baseline, this model achieves up to 27% reduction in excess volatility at P50 and 7% at P90. By also adding decoder-self attention we see a further reduction in excess volatility of an additional 20% at P50 and 21% at P90.

[2] Wen et al. (2017), Figure 3 shows MQ-CNN (labeled "MQ_CNN_wave") outperforms MQ-RNN (all variants) and DeepAR (labeled "Seq2SeqC") on the test set.

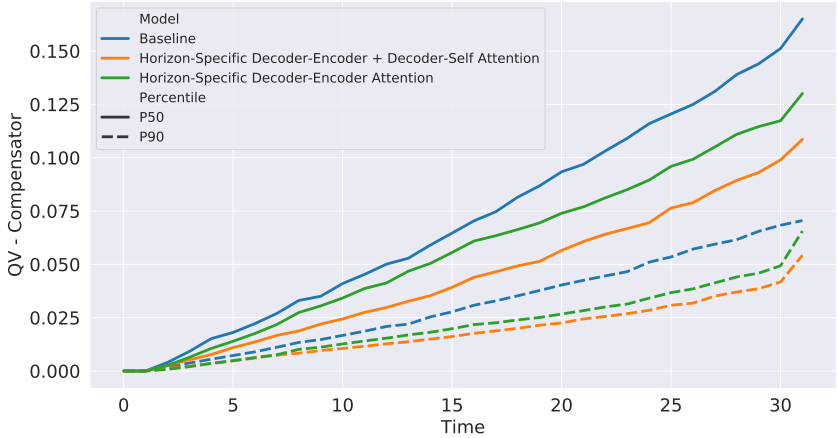

Figure 2: Martingale diagnostic process $\{V_t\}$ averaged over all weeks in test period (2018-2019)

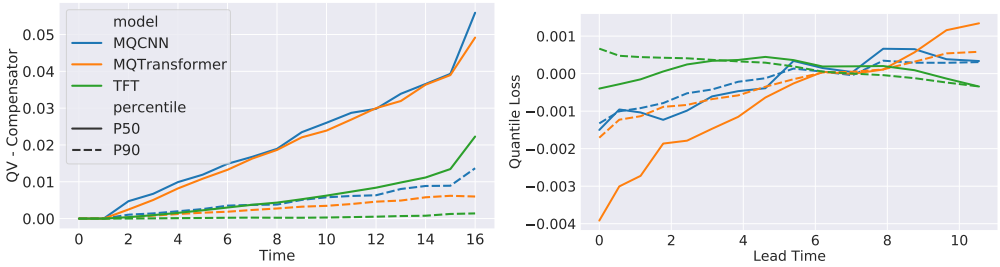

Figure 3: Forecast evolution analysis on the retail dataset. Left: Martingale Diagnostic Process $\{V_t\}$. Right: QL by lead time, averaged over target dates from 2016-03-01 through 2016-05-01; QL trajectories are centered around 0.

4.2 Publicly Available Datasets

Following Lim et al. (2019), we consider applications to brick-and-mortar retail sales, electricity load, securities volatility and traffic forecasting. For the retail task, we predict the next 30 days of sales, given the previous 90 days of history. This dataset contains 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 additional information about these tasks. Table 4 compares MQTransformer's performance with other recent works[3] – DeepAR (Salinas et al., 2020), ConvTrans (Li et al., 2019), MQ-RNN (Wen et al., 2017), and TFT (Lim et al., 2019).

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 4. 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 (Lim et al., 2019) – see Appendix D. 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 4. We can observe that MQTransformer still dramatically outperforms TFT, and its performance is similar to the MQ-CNN baseline trained on all trajectories.

Furthermore, on the Favorita retail forecasting task, Figure 3 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 3, 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*

[3]Results for TFT, MQ-RNN, DeepAR and ConvTrans are from Lim et al. (2019). For MQ-CNN and MQTransformer, we use their pre-processing and evaluation code to ensure parity

Table 4: P50 (50th percentile) and P90 (90th percentile) QL on electricity and retail datasets with the best results on each task emphasized. For the retail task, MQTransformer has results in parentheses, which correspond to training without forking sequences on 450K trajectories only.

	P50 QL					
TASK	DEEPAR	CONVTRANS	MQ-RNN	MQ-CNN	TFT	MQTRANSFORMER
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
	P90 QL					
TASK	DEEPAR	CONVTRANS	MQ-RNN	MQ-CNN	TFT	MQTRANSFORMER
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**

accurate as it fails to incorporate newly available information. By contrast, MQTransformer is both *less volatile* and *more accurate* when compared with MQ-CNN. See Appendix D for more details on the experiment setup and training procedure used.

Computational Efficiency For purposes of illustration, consider the Retail dataset. Prior work (Lim et al., 2019) was only able to make use of 450K out of 20M trajectories, and the optimal TFT architecture required 13 minutes per epoch (minimum validation error at epoch 6)[4] using a single V100 GPU. By contrast, our innovations are compatible with forking sequences, and thus our architecture can make use of *all available trajectories*. To train, MQTransformer requires only 5 minutes per epoch (on 20M trajectories) using a single V100 GPU (minimum validation error reached after 5 epochs). Of course, some of the differences in runtime can be attributed to use of different deep learning frameworks, but it is clear that MQTransformer (and other MQForecasters) can be trained much more efficiently than models like TFT and DeepAR.

5 CONCLUSIONS AND FUTURE WORK

In this work, we present three novel architecture enhancements that improve bottlenecks in state of the art MQ-Forecasters. We presented a series of architectural innovations for probabilistic time-series forecasting including a novel alignment decoder-encoder attention, as well as a decoder self-attention scheme tailored to the problem of multi-horizon forecasting. To the best of our knowledge, this is the first work to consider the impact of model architecture on forecast evolution. We also demonstrated how position embeddings can be learned directly from domain-specific event indicators and horizon-specific contexts can improve performance for difficult sub-problems such as promotions or seasonal peaks. Together, these innovations produced significant improvements in the excess variation of the forecast and accuracy across different dimensions. Finally, we applied our model to several public datasets, where it outperformed the baseline architecture by 5% at P50, 11% at P90 and the previous reported state-of-the-art (TFT) by 38% on the most complex task. On the three less complex public datasets, our architecture achieved parity or slightly exceeded previous state of the art results. Beyond accuracy gains, by making our architecture innovations compatible with forking sequences, our model achieves massive increases in throughput compared to existing transformer architectures for time-series forecasting. An interesting direction we intend to explore in future work is incorporating encoder self-attention so that the model can leverage arbitrarily long historical time series, rather than the fixed length consumed by the convolution encoder.

[4]Timing results obtained by running the source code provided by Lim et al. (2019)

REFERENCES

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

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

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.

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

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.

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

CINAR, Y. G., MIRISAEE, H., GOSWAMI, P., GAUSSIER, E., AIT-BACHIR, A. and STRIJOV, V. (2017). Position-based content attention for time series forecasting with sequence-to-sequence rnns. In *ICONIP*.

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*.

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

GALASSI, A., LIPPI, M. and TORRONI, P. (2019). Attention, please! a critical review of neural attention models in natural language processing. arXiv:1902.02181.

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

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.

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.

KIM, S. and KANG, M. (2019). Financial series prediction using Attention LSTM. arXiv:1902.10877.

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

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*.

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

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

MADEKA, D., SWINIARSKI, L., FOSTER, D., RAZOUMOV, L., TORKKOLA, K. and WEN, R. (2018). Sample path generation for probabilistic demand forecasting. In *ICML workshop on Theoretical Foundations and Applications of Deep Generative Models*.

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

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.

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

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

SHUN-YAO SHIH AND FAN-KENG SUN AND HUNG-YI LEE (2019). Temporal pattern attention for multivariate time series forecasting. *Machine Learning* **108** 1421–1441.

STINE, R. and FOSTER, D. (2020a). Martingale Descriptions of the Evolution of Forecasts. Tech. rep., Manuscript in preparation.

STINE, R. and FOSTER, D. (2020b). Martingale Diagnostics. Tech. rep., Manuscript in preparation.

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

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

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

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.

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*.

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

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

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

XU, K., BA, J., KIROS, R., CHO, K., COURVILLE, A., SALAKHUDINOV, R., ZEMEL, R. and BENGIO, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *ICML*.

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 sth 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. (2017) uses a sinusoidal positional encoding added to the input to an attention block, providing each token's position in the input time series.

B MQTransformer Architecture Details

In this section we describe in detail the layers in our MQTransformer architecture, which is based off of the MQ-Forecaster framework (Wen et al., 2017) and uses a wavenet encoder (van den Oord et al., 2016) for time-series covariates. Before describing the layers in each component, Figure 4 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 layer with rate p_{drop}, and the local position encoding is implemented as a single dense layer of dimension d_h.

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 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 1 for a description of the blocks in the decoder. The dimension of each head's in both the horizon-specific and decoder self-attention blocks is $\lceil d_h/8 \rceil$. The dense layer used to compute $\boldsymbol{c}_t^a$ has dimension d_h. The output block is two layer MLP with hidden layer dimension d_h, and weights are shared across all time points and horizons. The output layer has one output per horizon, quantile pair.

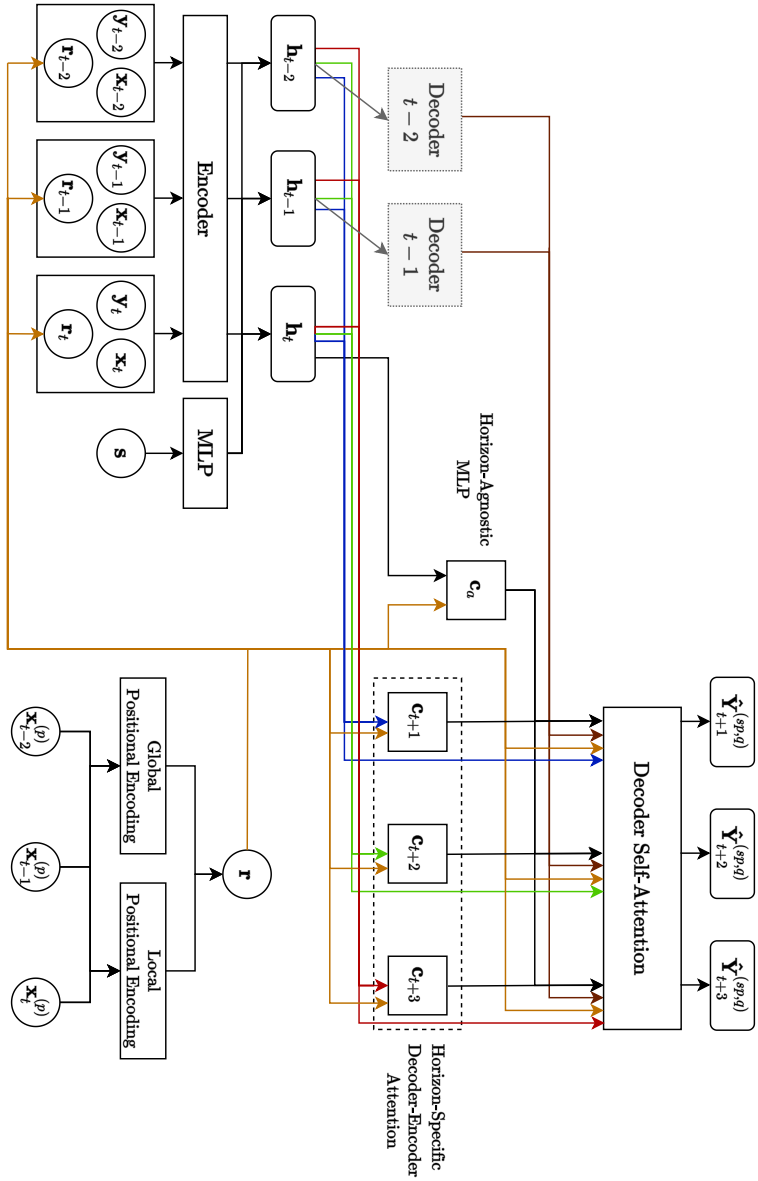

Figure 4: MQTransformer architecture with learned global/local positional encoding, horizon-specific decoder-encoder attention, and decoder self-attention

C LARGE SCALE DEMAND FORECASTING EXPERIMENTS

C.1 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 5.

Table 5: 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	64
Attention Block Heads	52
Attention Block Head Dimension	8
Dropout Rate	0.15
Activation Function	ReLU

C.2 RESULTS

Tables 6, 7, and 8 contain the full set of results on the large scale demand forecasting task. Per the discussion in Section 4, we use MQ-CNN as the baseline model and we did not compare to TFT due to scaling issues and the fact that on the public task that was most similar (retail), MQ-CNN significantly outperformed TFT.

Quantile loss by horizon Table 6 demonstrates how the attention mechanism yields significant improvements in shorter LTSP (e.g. LTSP 3/1 and LTSP 0/4), 7% improvement in P90 QL for LTSP 3/1 and 7.6% improvement in P90 QL for LTSP 0/4. We still see improvements for longer LTSP, but they are less substantial: 3.8% improvement in P90 QL for LTSP 12/3 and 3.9% improvement in P90 QL for LTSP 0/33. By also adding decoder-self attention, we continue to see improved results for shorter LTSP compared to only decoder-encoder attention, but we do see slight degradations for longer LTSP when comparing to decoder-encoder attention.

Promotions Performance In Table 8 we see that MQTransformer outperforms the prior state of the art on all promotion types. After adding the horizon-specific decoder-encoder and decoder-self attentions, versus the baseline, we see a 49% improvement for Promotion Type 1 products, a 31% improvement for Promotion Type 2 products, and a 17% improvement for Promotion Type 3 products.

Peak Performance Table 3 illustrates that while MQCNN performs well on some seasonal peaks, it also is misaligned and fails to rampdown post-peak – ramp-down issues occur when the model continues to forecast high for target weeks after the peak week. By including MQTransformer's attention mechanisms in the architecture, we see a 43% improvement for Seasonal Peak 1, a 21%

improvement for Seasonal Peak 2, a 7% improvement for Seasonal Peak 3, and a 56% improvement on Post-Peak Rampdown.

Table 6: 52-week aggregate quantile loss metrics with for a set of representative lead times and spans

MODEL	ALL LTSP		LTSP 3/1		LTSP 0/4	
	P50	P90	P50	P90	P50	P90
BASELINE	1.000	1.000	1.000	1.000	1.000	1.000
DEC-ENC	**0.984**	**0.960**	0.950	0.927	0.963	0.931
DEC-ENC & DEC-SELF	0.989	0.984	**0.934**	**0.911**	**0.948**	**0.908**

MODEL	LTSP 12/3		LTSP 0/33	
	P50	P90	P50	P90
BASELINE	1.000	1.000	1.000	1.000
DEC-ENC	0.975	**0.957**	**0.982**	**0.963**
DEC-ENC & DEC-SELF	**0.960**	0.964	0.982	0.981

Table 7: P90 quantile loss metrics on seasonal peak target weeks

MODEL	SEASONAL PEAK 1	SEASONAL PEAK 2	SEASONAL PEAK 3	POST-PEAK RAMPDOWN
BASELINE	1.000	1.000	1.000	1.000
DEC-ENC	0.748	**0.817**	0.962	0.712
DEC-ENC + DEC-SELF	**0.698**	0.826	**0.931**	**0.639**

Table 8: P90 quantile loss metrics on item, weeks with promotions

MODEL	PROMOTION TYPE 1	PROMOTION TYPE 2	PROMOTION TYPE 3
BASELINE	1.000	1.000	1.000
DEC-ENC	0.706	0.769	0.865
DEC-ENC + DEC-SELF	**0.670**	**0.763**	**0.851**

D EXPERIMENTS ON PUBLIC DATASETS

In this section we describe the experiment setup used for the public datasets in Section 4.

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. (2019) for more details.

RETAIL

This dataset is provided by the Favorita Corporacion (a major Grocery chain in Ecuador) as part of a Kaggle[5] 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 forecast creation time. While Lim et al. (2019) 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 3, we used a 60 day validation window (March 1, 2016 through May 1, 2016), which corresponds to 30 forecast creation times.

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 Kingma and Ba (2015) 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.

[5]The original competition can be found here.

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 4.

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 Reported Model Parameters

The optimal parameters for each task are given in Table 9.

Table 9: 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]