MQ-ReTCNN: Multi-Horizon Time Series Forecasting with Retrieval-Augmentation

Sitan Yang
Forecasting Science, Amazon
New York, New York, USA
sitanyan@amazon.com

Carson Eisenach
Forecasting Science, Amazon
New York, New York, USA
ceisen@amazon.com

Dhruv Madeka
SCOT Reinforcement Learning, Amazon
New York, New York, USA
maded@amazon.com

ABSTRACT

Multi-horizon probabilistic time series forecasting has wide applicability to real-world tasks such as demand forecasting. Recent work in neural time-series forecasting mainly focus on the use of Seq2Seq architectures [27]. For example, MQTransformer [10] – an improvement of MQCNN [30] – has shown the state-of-the-art performance in probabilistic demand forecasting. In this paper, we consider several methods to enhance model performance by incorporating cross-entity information and propose adding a cross-entity attention mechanism along with a retrieval mechanism to select which entities to attend over. We demonstrate how our new model, MQ-ReTCNN, leverages the encoded contexts from a pretrained MQCNN model on the entire population to improve forecasting accuracy. Using MQCNN as our baseline model (due to computational constraints, we do not use MQTransformer), we first show on a small demand forecasting dataset that it is possible to achieve ∼3% improvement in test loss by adding a cross-entity attention mechanism where we attend over all other entities in the population. We then evaluate the model with our proposed retrieval mechanism – as a means of approximating an attention over a large population – on a large-scale demand forecasting application with over 2 million products and observe ∼1% performance gain over the MQCNN baseline.

CCS CONCEPTS

• **Time Series → Forecasting**.

KEYWORDS

Time Series, Multi-horizon Forecasting, Cross-Entity Attention, Retrieval-Augmentation

ACM Reference Format:
Sitan Yang, Carson Eisenach, and Dhruv Madeka. 2022. MQ-ReTCNN: Multi-Horizon Time Series Forecasting with Retrieval-Augmentation. In *MileTS '22: 8th KDD Workshop on Mining and Learning from Time Series, August 15th, 2022, Washington DC*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/1122445.1122456

MileTS '22, August 15th, 2022, Washington DC

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Multi-horizon probabilistic time series forecasting has many important applications in real-world tasks [5, 6, 10, 20, 23, 30]. For example, consider a retailer who wishes to optimize their purchasing decisions. In order to make optimal decisions, they require forecasts of consumer demand at multiple time steps in the future. In the domain of multi-horizon, probabilistic time-series forecasting, deep neural networks (DNNs), especially those of the Seq2Seq variety [27], have increasingly been studied [10, 11, 21, 25, 30]. They have various advantages over traditional time series models including the ability to easily handle a complex mix of historic covariates and the potential to incorporate recent advances in Seq2Seq learning.

Like many other machine learning tasks, the canonical formulation considers time-series for N entities – e.g. N products in the case of demand forecasting – and forecasts are produced using features specific to that entity. Models are trained using shared weights for each entity. Seq2Seq architectures consist of an *encoder*, which typically summarizes time-series covariates for an entity i into time specific representations, whch we denote as $h_{i,t}$ and a *decoder* which takes the encoded context and produces the output sequence (in this case, probabilistic forecasts). For an entity i and time t, we expect $h_{i,t}$ to be more relevant to the forecast target in inference than $h_{j,t}$ for any other $j \neq i$. That does not mean, however, that the encoded contexts of other entities contain no relevant information. As an example, consider forecasting demand for soda from two competing brands (brand A and brand B) – these products may be substitutes, and when the demand goes down for one, the other increases. In this way, information from brand A may be useful for forecasting for brand B, and vice-versa. Traditional Seq2Seq neural architectures such as RNN (Recurrent Neural Network), LSTM (Long Short-Term Memory network) and CNN (Convolution Neural Network) fail to capture such *cross-entity information*.

In natural language processing (NLP), a recent advance is retrieval-based language models [4, 13, 15] that directly search and utilize information from a large corpus, such as Wikipedia, to help inform predictions. For example, Retrieval-Enhanced Transformer (RETRO) introduced in [4] imprtoves language model performance not by scaling up model parameters or training data size, but via learning on information retrieved from a task-related database. The key idea is to apply attention [2] over representations of other entities in the population. Because the size of the population can be quite large, the authors propose using a k-nearest neighbors (k-NN) search similar to [15] to lookup up the most relevant entities and attend only over those.

Inspired by these studies, we introduce a cross-entity attention mechanism along with a retrieval mechanism to the state-of-the art

MQ-Forecaster framework [10, 29, 30] for probabilistic time series forecasting. In particular, we build this work on the MQCNN model [30] but our methods can be naturally extended to any generic Seq2Seq time series forecaster. For retrieval methods, in addition to the commonly used k-NN search, we also propose using an arbitrary submodular function to select a relevant set of entities to attend over and motivate the use of a submodular function to approximate an attention mechanism.

Our work is one of the first architectures to leverage cross-entity information with retrieval-augmentation, and to the best of our knowledge, is the first to do so in the domain of time-series forecasting. Our main contributions are the following:

(1) MQ-ReTCNN – a retrieval-based Seq2Seq architecture for multi-horizon time series forecasting. The model builds on the encoder-decoder architecture of MQCNN, and utilizes an offline database of entity representations which the model attends over during training and inference. We also incorporate a retrieval mechanism to efficiently select which entities to attend over so that the methodology can scale to large datasets. We show that our model brings noticeable accuracy gains over the MQCNN baseline on both a small-scale and large-scale demand forecasting problem.
(2) A new retrieval method that uses a submodular scoring function to efficiently summarize all contexts from the offline database rather than searching for nearest neighbors of each example as commonly used in the literature. As we show in the results section, this method achieves comparable performance to the k-NN search in our applications.

The rest of the paper is organized as follows: in Section 2, we provide an overview of the multi-horizon time series forecasting problem and related work. In Section 3 we describe our proposed methods in detail. In Section 4 we present the experimental results. We show that on our target application – demand forecasting – it is possible to achieve a 3% improvement over the baseline when attending over all other entities in the population on a small dataset (approximately 10K products). We then evaluate several retrieval mechanisms that scale the model to a much larger population (around 2M products) and allow us to obtain an improvement of approximately 1% over the baseline.

2 BACKGROUND AND RELATED WORK

2.1 Time-Series Forecasting

We consider the high-dimensional regression problem with a mix of inputs where at each time t and for each entity i, we forecast the distribution of y over the next H periods:

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

where $y_{i,\cdot}$ denotes the target time series of entity i, $x^{(h)}_{i,:t}$ are historic covariates up through time t, $x^{(f)}_{i,t:}$ are covariates that are known apriori (such as calendar information), and $x^{(s)}_{i}$ are static covariates.

Many recent works [10, 21, 30] have considered this forecasting problem. In this paper, our application of interest is demand forecasting for a large e-commerce retailer and downstream applications require only specific quantiles, not the full distribution. Accordingly, we focus on producing quantile forecasts similar to other recent works [10, 21, 30]. Our model architecture builds off of the MQCNN architecture introduced in [30].

2.2 Attention Mechanisms

Attention mechanisms [2, 12] compute 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 $\text{score}(\mathbf{u}, \mathbf{v}) := \mathbf{u}^\top \mathbf{v}$. Often, one takes $\mathbf{q}_s = \mathbf{k}_s = \mathbf{v}_s = \mathbf{h}_s$, where $\mathbf{h}_s$ is the hidden state at time s.

The transformer architecture was first proposed in [28] and achieved state-of-the-art performance in language modeling. In the vanilla transformer, 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 $\mathbf{q}_s$ and key $\mathbf{k}_t$ is defined as follows

$$A^h_{s,t} = \mathbf{q}_s^\top \mathbf{W}_q^{h,\top} \mathbf{W}_k^h \mathbf{k}_t. \tag{2}$$

This architecture design has been successfully adopted in many subsequent studies with various extensions such as Transformer-XL [7], Reformer [17] and most recently Retrieval-Enhanced Transformer [4].

2.3 Retrieval Mechanisms

Information retrieval is a classic topic for language modeling and a recent advance is the retrieval-based models. Several latest works have demonstrated the benefit of adding an explicit retrieval step to neural networks. In [15], kNN-LM is proposed to enhance a language model through a nearest neighbor search in suitable text collections. [13] introduces REALM which augments language model pretraining with a latent knowledge retriever. More recently, RETRO [4] enhances the model architecture not by increasing the number of parameters or the size of training data, but rather through the retrieval of information relevant for each sample. Similarly, [3] uses memorized similarity information from the training data for retrieval at at inference time.

2.4 Data Summarization and Submodular Functions

Data summarization has gained a lot of interest in recent years with the application of so called *Submodular Functions*. Applications range from exemplar-based clustering [9] to document summarization [8, 22]. The goal is to select representative subsets of elements from a large-scale dataset through a pre-defined optimization process. The key component of the optimization formulation is a submodular function which serves as a scoring function for any particular subset.

Definition 1 (Submodular Function). *Let Ω be a finite set. A function $f : 2^\Omega \to \mathbb{R}$ is said to be submodular if for any $S \subseteq T \subseteq \Omega$ and any $x \in \Omega \setminus S$*

$$f(S \cup \{x\}) - f(S) \geq f(T \cup \{x\}) - f(T).$$

The essential property of submodular functions is known as submodularity, an intuitive diminishing returns condition that allows the search for nearly-optimal solutions in linear time and fits well into the purpose of subset selection. The formal definition is given as Definition 1 and we direct the reader to [18] for a thorough overview of submodular functions and their optimization. Many recent applications of submodular optimization focus on scaling up traditional algorithms to dealing with massive amounts of data or data streams. Proposed methods include distributed algorithms [19, 24] and streaming algorithms [1].

3 METHODOLOGY

3.1 Problem Formulation

As mentioned in Section 2, we aim to estimate the distribution of the target variable y_i as presented in Equation (1) over the next H horizons at each time t. We train a quantile regression model to minimize the total *quantile loss*, summed over all forecast creation times (FCTs) T with Q quantiles and H horizons

$$\sum_t \sum_q \sum_h L_q \left(y_{i,t+h}, \widehat{y}^{(q)}_{i,t+h} \right), \tag{3}$$

where $L_q(y, \widehat{y}) = q(y - \widehat{y})_+ + (1 - q)(\widehat{y} - y)_+$, $(\cdot)_+$ is the positive part operator, t denotes a FCT, q denotes a quantile, and h denotes the horizon. In this paper, we adopt the multi-horizon forecasting setting described in [10, 21, 30] with the output of the 50th and 90th percentiles (P50 and P90) at each time step, and thus the model is trained to jointly minimize the P50 and P90 quantile loss.

3.2 Model Architecture

We design our model to be capable of leveraging an offline database constructed using the encoded representations from a frozen, pre-trained base model. In this paper, we use MQCNN [30] as the base architecture rather than the state-of-the-art MQTransformer [10] as the latter one requires substantially more GPU memory and, as discussed below, we are already memory bound. Further, we expect that our retrieval mechanism offers an improvement that is orthogonal to those in MQTransformer, and the two sets of improvements could be combined in future work.

Generally our model adopts the Seq2Seq structure of MQCNN with an encoder that produces an encoded context at time t

$$h_{i,t} := \text{encoder}(y_{i,:t}, x^{(h)}_{i,:t}, x^{(s)}_i),$$

and a decoder that differs from MQCNN in that we include an additional "cross-entity context", which we denote as $\widetilde{h}_{i,t}$. Formally, the decoder computes

$$\widehat{\mathbf{Y}}_{i,t} := \text{decoder}(h_{i,t}, \widetilde{h}_{i,t}, x^{(f)}_{i,t:})$$

where $\widehat{\mathbf{Y}}_{i,t}$ is a matrix of shape $H \times Q$ for forecast quantiles of different horizons. We also denote $\mathbf{H} := \{h_{i,t} | \forall i, t\}$ and $\widetilde{\mathbf{H}} := \{\widetilde{h}_{i,t} | \forall i, t\}$. Ideally, $\widetilde{\mathbf{H}}$ would be computed by attending *all other* entities in the database, but for large datasets this may become infeasible. Thus we add a retrieval mechanism to select an informative subset of entities to attend across at each time step, which we provide more details in the next section.

To generate $\widetilde{\mathbf{H}}$, we add a *time series cross-attention* layer after the encoder to extract the cross-entity information through attention between the retrieved contexts and examples during training. The attention is computed only at each time step across different entities, and no cross time (temporal) attention is currently considered. Proper masking is used to make sure the attended and attending entities are aligned as shown in Figure 2.

We find in our experiments that this process increases the GPU memory consumption of the model because the retrieved contexts are loaded with each mini-batch during training. This in turn limits the total number of elements contained in these contexts, which makes the retrieval mechanism necessary on large datasets.

The overall architecture of our model, MQ-ReTCNN, is depicted in Figure 1, and we adopt a similar mechanism to incorporate cross-entity information for NLP tasks as shown in Figure 2 of [4].

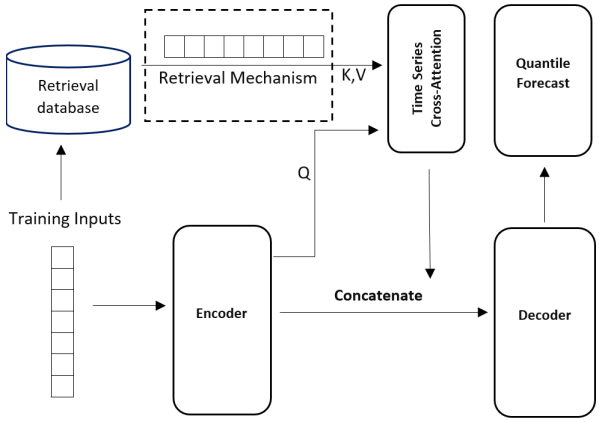

Figure 1: An overview of the MQ-ReTCNN architecture; adapted from [4].

3.3 Retreival Mechanisms

In this paper, retrieval mechanisms play a key role in scaling up the model to large datasets. In particular, we denote the offline database as $\mathbf{H}^0 := \{h^0_{i,t} | \forall i, t\}$ which consists of the encoded contexts produced by a pre-trained MQCNN encoder. The retrieval calculations are only based on $\mathbf{H}^0$ to determine which entities for each example to attend over during training.

Broadly, we consider two types of retrieval mechanisms:

(1) Entity-specific retrieval of relevant entities defined as nearest neighbors.
(2) A shared set of entities from the population that are "maximally relevant" and used to produce the cross entity context for each entity.

See Figure 2 for a visualization of the two different approaches.

Entity-Specific Nearest Neighbors For each entity i, we consider searching for the nearest nearest neighbors in our offline database. This can be formulated as:

$$\underset{S;|S|=K}{\text{argmax}} \sum_{j \in S, j \neq i} f(h^0_{i,t}, h^0_{j,t}), \quad \forall i, t. \tag{4}$$

Here we find a set of K elements that maximize some similarity metric between example i and elements in S, and we search for such

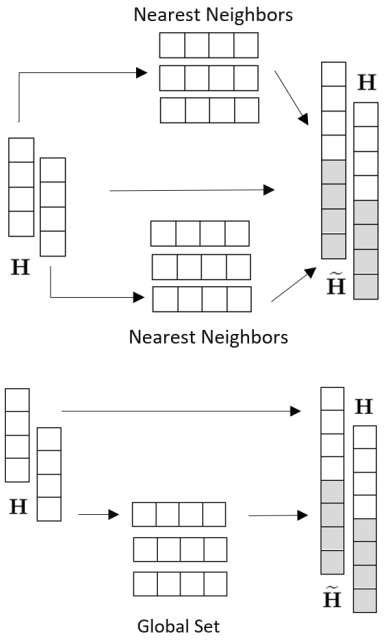

Figure 2: The top diagram depicts nearest neighbor retrieval, the bottom digram depicts retrieval using a shared global set. The output of the retreival step is concatenated to the input embedding vector.

set at each time step t; that is, we find a time-specific set of k nearest neighbors. We can take $f(\cdot,\cdot)$ to be any similarity metric, and in this paper we consider the Pearson correlation – which is essentially equivalent to the dot-product attention – and is computed as

$$f(h^0_{i,t}, h^0_{j,t}) = \frac{< h^{0,c}_{i,t}, h^{0,c}_{j,t} >}{\|h^{0,c}_{i,t}\| \|h^{0,c}_{j,t}\|}$$

where $h^{0,c}_{j,t}$ denotes the centered version of $h^0_{j,t}$.

Global Set via Submodular Maximization Denote by $L(\theta; S)$ the loss in Equation (3) evaluated for a model with parameters θ and set of S of entities to attend over, and let $\mathcal{A}$ denote the set of all entities. We would like to select a set S of size K such that

$$\underset{S \subseteq \mathcal{A}: |S| \leq K}{\operatorname{argmin}} \min_{\theta} L(\theta; S).$$

Solving the outer minimization above is not tractable, so instead we consider using a submodular proxy objective to select the set S. Specifically we formulate this as follows for each time t:

$$\underset{S; |S|<k}{\operatorname{argmax}} \sum_{i \in \mathcal{A}} \max_{j \in S, j \neq i} f(h^0_{i,t}, h^0_{j,t}). \tag{5}$$

Here we use the same similarity metric f as in Equation (4). In general, the form of Equation 5 is referred to as the *Facility Location* problem in [18], and it is a classic example of optimizing submodular function.

Time-Specific vs. Time-Agnostic Retrieval

Equation (4) and (5) require the retrieval calculation to be carried out at each time step, both for model training and inference. The advantage is that the retrieval process can then be adaptive to each time step (e.g., nearest neighbors can be different for each time step) and is performed on-the-fly at the inference time. But it can be computationally expensive. Alternatively we propose using $v^0_i := \sum_{t=1}^{T} h^0_{i,t}$ instead of $h^0_{i,t}$ as follows

$$\underset{S; |S|=k}{\operatorname{argmax}} \sum_{j \in S, j \neq i} f(v^0_i, v^0_j) \tag{6}$$

$$\underset{S; |S|<k}{\operatorname{argmax}} \sum_{i \in \mathcal{A}} \max_{j \in S, j \neq i} f(v^0_i, v^0_j) \tag{7}$$

i.e., we define the retrieved set S to be *time-agnostic* by considering all time steps in the training window rather than *time-specific* as done previously. In this case, we use the exact same set of entities (but with different contexts for the test period) for model inference and no more retrieval calculation is needed. Note that this does not lead to any information leakage as no computation is done on the test set. We compare the performance of these two types of retrieval mechanisms in the next section.

4 RESULTS

In this section we evaluate on a large demand forecasting dataset using two different experimental setups. The dataset comes from a large e-commerce retailer and includes time series features such as demand, promotions, holidays and detail page views as well as static metadata features such as catalog information. Similar datasets with the same set of features but generated in different time windows have been used in [10, 30]. Here we have four years (2015-2019) of data for approximately over 2 million products. The task is to forecast the 50th and 90th quantiles of demand for each of the next 52 weeks at each forecast creation time t.

Each model is trained using up to 8 NVIDIA V100 Tensor Core GPUs, on three years of data (2015-2018) and one year is held out for evaluation (2019). In the "small scale" setup, we consider only 10,000 different products (entities) so that we can directly attend over a representation of all products rather than use any retrieval method. In the "large scale" setup, we have too many to directly attend over all of them simultaneously. Instead, we demonstrate our model can scale up to the entire dataset using retrieval methods, which we ablate and compare the resulting model performance.

4.1 Small Scale

In this experiment we choose the 10K products with the largest total units sold during the training period, and we compare four different architectures:

- MQCNN: baseline MQCNN model
- MQCNN-L: MQCNN with the increased model capacity
- MQRet-Full: MQCNN with cross entity context $\widetilde{h}_{i,t}$ produced by attending the frozen context across *all other* entities at time t (i.e. from the database $\mathbf{H}^0$).
- MQRet-Random: Same as above, but where all $\widetilde{h}_{i,t}$ are randomly generated.

By comparing MQRet-Full with other models, we can better understand how much improvement is possible by augmenting the model with the cross-entity context generated from the entire population. We include two ablations to confirm that the improvement in performance is due to extracting useful cross-entity information. In particular, for testing whether increasing model capacity can lead to performance gain, we consider MQCNN-L which expands MQCNN's capacity by increasing the number of filters of the CNN layer, so that MQRet-Full and MQCNN-L have the same number of parameters. We also consider MQRet-Random, which has the same architecture as MQRet-Full but with randomly generated (non-informative) contexts. We train each model to 100 epochs using batch size of 256, and optimize using ADAM [16]. Table 1 gives the number of parameters in each trained model.

Table 1: The number of parameters used in the four different architectures of the small scale experiment.

Model	Number of Parameters
MQCNN	0.86×10^6
MQCNN-L	1.22×10^6
MQRet-Random	1.21×10^6
MQRet-Full	1.21×10^6

Table 2: Experiment results on 10K products. All results are rescaled so they are relative improvements over the baseline MQCNN model, lower is better.

Model	P50	P90	Overall
MQCNN	1.000	1.000	1.000
MQCNN-L	0.990	0.996	0.993
MQRet-Random	1.008	1.007	1.008
MQRet-Full	0.968	0.978	0.973

Table 2 shows the (rescaled) quantile loss results (P50, P90 and overall) for the four models described above. We calculate these results based on three different runs of each model and average the performance metrics. As expected, we observe no accuracy gains from MQRet-Random, as there is no signal to extract. MQCNN-L yields very slight improvement by simply increasing the model capacity. By contrast, MQRet-Full brings relatively substantial improvements in overall performance, improving P50 by 3.2% and P90 by 2.2%. Thus, the model seems to be extracting useful signal from other entities.

4.2 Large Scale

For this experiment, we use the whole dataset of over 2 million products. The training and test split is kept the same as in the first experiment.

We evaluate the MQ-ReTCNN architecture in Figure 1 with both retrieval mechanisms described previously, and consider both time-specific and time-agnostic variants. For the nearest neighbor method, we use FAISS [14], an open source library for fast nearest neighbor retrieval in high dimensional spaces, and we set $k = 10$. For the submodular method, we use Apricot [26] which provides efficient submodular optimization tools. In this case, we choose $k = 10000$ for the size of the global set. We selected these values for K to maximize utilization of available GPU memory.

Overall, we consider the following MQRet model variants:

- MQCNN: baseline MQCNN model
- MQRet-KNN: MQ-ReTCNN with time-agnostic, nearest neighbor retrieval.
- MQRet-Subm: MQ-ReTCNN with time-agnostic, submodular retrieval.
- MQRet-KNN-t: MQ-ReTCNN with time-specific, nearest neighbor retrieval.
- MQRet-Subm-t: MQ-ReTCNN with time-specific, submodular retrieval.

Table 3: Experiment results on the whole dataset. Results are rescaled so they are relative improvements over the baseline MQCNN model, lower is better.

	Model	P50	P90	Overall
All Horizons	MQCNN	1.000	1.000	1.000
	MQRet-KNN	0.999	0.973	0.987
	MQRet-KNN-t	0.991	0.986	0.989
	MQRet-Subm	0.993	0.996	0.994
	MQRet-Subm-t	0.991	0.988	0.990
$h \leq 10$	MQCNN	1.000	1.000	1.000
	MQRet-KNN	0.995	0.971	0.984
	MQRet-KNN-t	0.986	0.993	0.989
	MQRet-Subm	0.983	0.989	0.986
	MQRet-Subm-t	0.985	0.991	0.985

We train each model for 100 epochs with a batch size of 512. Test results are summarized in Table 3. We include the model performance aggregated across all horizons (52 weeks) as well as for horizons $h \leq 10$. We observe that all MQRet variants improve the overall performance by around 1% but the gains are smaller than the full cross-entity attention in Table 2. Larger performance improvements are observed for all models when aggregated over shorter horizons. The performance of time-specific models are generally similar to that of time-agnostic ones. The best variant – MQRet-KNN – improves by 1.3% over the baseline MQCNN model for all horizons, and by 1.5% when restricted to only shorter horizons ($h \leq 10$).

5 CONCLUSION

In this paper we demonstrated that incorporating cross-entity information can improve the predictive accuracy of time-series forecasting models. On our target application, we showed approximately a 3% improvement over the baseline model when we attended over all other entities in the population. The gains on the large scale dataset were smaller – approximately over 1% improvement on the baseline. Accordingly, a future directions of interest is training a model that can attend across all entities during each forward pass –

will require model parallelism across multiple machines. Another interesting direction of future inquiry is using pretrained graphs between entities to select the nearest neighbors.

REFERENCES

[1] Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. 2014. Streaming Submodular Optimization: Massive Data Summarization on the Fly. In *Proc. ACM Conference on Knowledge Discovery in Databases (KDD)*.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. arXiv:1409.0473

[3] Giovanni Bonetta, Rossella Cancelliere, Ding Liu, and Paul Vozila. 2021. Retrieval-Augmented Transformer-XL for Close-Domain Dialog Generation. https://doi.org/10.32473/flairs.v34i1.128369 arXiv:2112.04426

[4] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. 2021. Improving language models by retrieving from trillions of tokens. arXiv:2112.04426

[5] Joos-Hendrik Böse, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Dustin Lange, David Salinas, Sebastian Schelter, Matthias Seeger, and Yuyang (Bernie) Wang. 2017. Probabilistic demand forecasting at scale. In *VLDB 2017*.

[6] Carlos Capistrán, Christian Constandse, and Manuel Ramos-Francia. 2010. Multi-horizon inflation forecasts using disaggregated data. *Economic Modelling* 27, 3 (2010), 666–677.

[7] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context. In *ACL*. arXiv:1901.02860

[8] Anirban Dasgupta, Ravi Kumar, and Sujith Ravi. 2013. Summarization Through Submodularity and Dispersion. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers*.

[9] Delbert Dueck and Brendan J. Frey. 2007. Non-metric affinity propagation for unsupervised image categorization. *2007 IEEE 11th International Conference on Computer Vision* (2007), 1–8.

[10] Carson Eisenach, Yagna Patel, and Dhruv Madeka. 2020. MQTransformer: Multi-Horizon Forecasts with Context Dependent and Feedback-Aware Attention. arXiv:2009.14799

[11] Valentin Flunkert, David Salinas, and Jan Gasthaus. 2017. DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks. *CoRR* abs/1704.04110 (2017). arXiv:1704.04110 http://arxiv.org/abs/1704.04110

[12] Andrea Galassi, Marco Lippi, and Paolo Torroni. 2019. Attention, please! a critical review of neural attention models in natural language processing. arXiv:1902.02181

[13] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: Retrieval-Augmented Language Model Pre-Training. *CoRR* abs/2002.08909 (2020). arXiv:2002.08909

[14] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with GPUs.

[15] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2019. Generalization through Memorization: Nearest Neighbor Language Models. *CoRR* abs/1911.00172 (2019). arXiv:1911.00172

[16] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

[17] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The Efficient Transformer. arXiv:2001.04451 https://arxiv.org/abs/2001.04451

[18] Andreas Krause and Daniel Golovin. 2014. Submodular Function Maximization. In *Tractability*.

[19] Ravi Kumar, Benjamin Moseley, Sergei Vassilvitskii, and Andrea Vattani. 2015. Fast Greedy Algorithms in MapReduce and Streaming. *ACM Trans. Parallel Comput.* 2, 3 (2015), 14:1–14:22. http://dblp.uni-trier.de/db/journals/topc/topc2.html#KumarMVV15

[20] Bryan Lim. 2018. Forecasting Treatment Responses Over Time Using Recurrent Marginal Structural Networks. In *Advances in Neural Information Processing Systems*, Vol. 31.

[21] Bryan Lim, Sercan O. Arik, Nicolas Loeff, and Tomas Pfister. 2019. Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting. arXiv:1912.09363

[22] Hui Lin and Jeff Bilmes. 2011. A Class of Submodular Functions for Document Summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

[23] Dhruv Madeka, Lucas Swiniarski, Dean Foster, Leo Razoumov, Kari Torkkola, and Ruofeng Wen. 2018. Sample path generation for probabilistic demand forecasting. In *ICML workshop on Theoretical Foundations and Applications of Deep Generative Models*.

[24] Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause. 2013. Distributed Submodular Maximization: Identifying Representative Elements in Massive Data. In *Advances in Neural Information Processing Systems*, C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger (Eds.), Vol. 26. Curran Associates, Inc.

[25] Kin G Olivares, Nganba Meetei, Ruijun Ma, Rohan Reddy, and Mengfei Cao. 2021. Probabilistic Hierarchical Forecasting with Deep Poisson Mixtures. *arXiv preprint arXiv:2110.13179* (2021).

[26] Jacob Schreiber, Jeffrey Bilmes, and William Stafford Noble. 2019. apricot: Submodular selection for data summarization in Python.

[27] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, , and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

[29] Ruofeng Wen and Kari Torkkola. 2019. Deep Generative Quantile-Copula Models for Probabilistic Forecasting. In *ICML Time Series Workshop*.

[30] Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. 2017. A multi-horizon quantile recurrent forecaster. In *NIPS Time Series Workshop*.