Multi-Horizon Time Series Forecasting with Temporal Attention Learning

Chenyou Fan[1], Yuze Zhang[1], Yi Pan[1], Xiaoyue Li[1], Chi Zhang[1], Rong Yuan[1], Di Wu[1]
Wensheng Wang[1], Jian Pei[2,3], Heng Huang[2,4*]
[1]JD.com [2] JD Digits
[3] School of Computing Science, Simon Fraser University
[4] Electrical & Computer Engineering, University of Pittsburgh

ABSTRACT

We propose a novel data-driven approach for solving multi-horizon probabilistic forecasting tasks that predicts the full distribution of a time series on future horizons. We illustrate that temporal patterns hidden in historical information play an important role in accurate forecasting of long time series. Traditional methods rely on setting up temporal dependencies manually to explore related patterns in historical data, which is unrealistic in forecasting long-term series on real-world data. Instead, we propose to explicitly learn constructing hidden patterns' representations with deep neural networks and attending to different parts of the history for forecasting the future. In this paper, we propose an end-to-end deep-learning framework for multi-horizon time series forecasting, with temporal attention mechanisms to better capture latent patterns in historical data which are useful in predicting the future. Forecasts of multiple quantiles on multiple future horizons can be generated simultaneously based on the learned latent pattern features. We also propose a multimodal fusion mechanism which is used to combine features from different parts of the history to better represent the future. Experiment results demonstrate our approach achieves state-of-the-art performance on two large-scale forecasting datasets in different domains.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; **Machine learning algorithms**; • **Applied computing** → **Supply chain management**; *E-commerce infrastructure*; Enterprise modeling.

KEYWORDS

deep learning, recurrent neural network, forecasting, attention model, supply chain forecasting

ACM Reference Format:
Chenyou Fan[1], Yuze Zhang[1], Yi Pan[1], Xiaoyue Li[1], Chi Zhang[1], Rong Yuan[1], Di Wu[1] and Wensheng Wang[1], Jian Pei[2,3], Heng Huang[2,4]. 2019. Multi-Horizon Time Series Forecasting with Temporal Attention Learning. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330662

*To whom all correspondence should be addressed.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
KDD '19, August 4–8, 2019, Anchorage, AK, USA
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330662

1 INTRODUCTION

Time series forecasting problem is to study how to predict the future accurately based on the historical observations. Increasing forecasting accuracy is beneficial to operational efficiency in many aspects of society. For example, data-driven demand forecasting techniques enable online retailers to gain a better understanding of market demands, thus improving supply chain operation efficiency such as delivery speed and product in-stock rate.

Classical time series forecasting approaches include Holt-Winters method [15, 30] and ARIMA [5]. These models are ineffective in modeling highly nonlinear time series. Recently, Recurrent Neural Networks (RNNs), and the frequently used variant Long Short-Term Memory Networks (LSTMs), have been proposed for modeling complicated sequential data, such as natural language [27], audio waves [25], and video frames [7]. LSTMs have also been applied to forecasting tasks in recent studies [4, 9], and have shown the ability to capture complex temporal patterns in dynamic time series. By using an LSTM encoder-decoder model to map the history of a time series to its future, multi-step forecasting can be naturally formulated as sequence-to-sequence learning. Prior models based on this architecture outperformed classical methods on several benchmark forecasting datasets, e.g. GEFCom2014 Electricity [16], but with shortcomings that this work addresses.

In many practical circumstances, multi-horizon forecasting (forecasting on multiple steps in future time) is preferred, as it provides guidance for resource scheduling and decision making over a period of time. This topic has received more and more attention in recent studies [6, 29]. However, forecasting on long horizons brings challenges to existing LSTM-based approaches in exploring temporal patterns on future horizons, where local dynamics stem from both historical information and dynamic input variables on future horizons. We will show that our method proposes to model proper latent representations of the future dynamics by referring to most relevant temporal patterns in historical data and thus produces accurate forecasts.

Moreover, sometimes it is necessary to forecast the overall distribution of the target to help business decision. An a typical example, inventory planning requires forecasts of sales of products with different levels of overestimation to reduce stock-out costs. Depending on the popularity and the stock-out cost of each product, different levels of overestimation are chosen from the forecast distributions. In these cases, quantile regression has been widely used to make

predictions of different quantiles to approximate the target distribution without making distributional assumptions. Mean regression, also known as the least squares method, is a special case of quantile regression which focuses on conditional mean, while quantile regression can extend to arbitrary quantile. For sales forecasting, the targeted quantile of a product can be interpreted as the "intolerance" of being out of stock, e.g., the target quantile of a high-end smartphone could be as high as 0.9 while that of a third-party cable could be only 0.5. In Fig 1, we give an example of sale forecasting with quantile predictions. Given the daily sales of 50,000 products from January to March, the target is to predict sales of the future, e.g., from April to June. Shaded blue areas demonstrate the quantile predictions between 0.2 and 0.8, which we observe that they are robust to sudden changes. Many recent forecasting models [29, 31] are designed to produce quantile estimations which are evaluated with quantile loss functions. We will show that our proposed model can make efficient quantile estimations without adding additional complex network structures.

In this paper we propose an end-to-end deep-learning framework for multi-horizon time series forecasting, with novel structures to better capture temporal patterns on future horizons. The idea is to first propagate information of future input variables in both forward and backward directions with a bi-directional LSTM decoder, considering dynamic future information such as promotions and calendar events. Then at each future time step we use the decoder hidden state to attend to several different periods of the history and generate attention vector individually. We treat different periods of the history as different modalities and we combine them by learning relative importance of each modality for predicting current time step. The combined feature, which we call temporal context feature, incorporates both historical information and future contextual information that can best describe current time step. We use a fully connected layer to emit quantile predictions based on the temporal context feature and the entire framework can be trained end-to-end and deployed with standard deep-learning platform such as Tensorflow [1] and PyTorch [26]. Our approach outperforms current state-of-the-art models on two large-scale forecasting datasets in different domains.

In the next section, we review related work on RNNs, quantile regression, and recent RNN-based approaches to forecasting. The following sections explain our network architecture including the encoder-decoder structure, temporal attention mechanism and multimodal fusion. Then we introduce two large-scale forecasting datasets - JD50K sales forecasting dataset and the GEFCom2014 electricity price forecasting dataset, and describe their features and targets in detail. Finally, we explain our experimental settings and demonstrate state-of-the-art results.

2 RELATED WORK

Neural Networks (NNs), including Convolutional Neural Networks (CNNs), have recently emerged as powerful models for many essential tasks in machine learning and computer vision [11, 21]. Recurrent Neural Networks (RNNs) are a special type of NNs which contain self-connections. Unlike feedforward NNs, the hidden states of RNNs serve as memory to help map both current time inputs

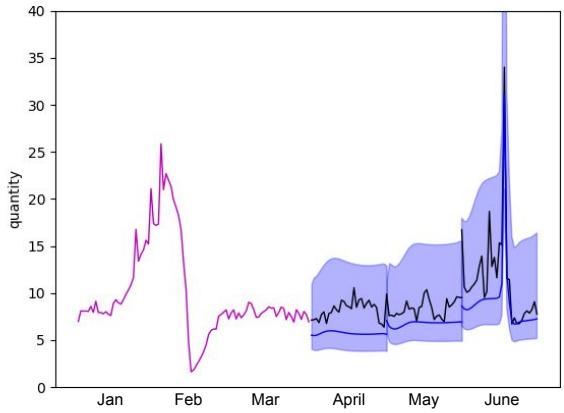

Figure 1: An example of sales forecasts on real-world online sales data with quantile predictions. The average daily sales of 50,000 products of each month are shown. Given historical daily sales from January to March 2018 as shown with magenta line, the task is to forecast from April to June. The black line shows the true sales; the dark blue line shows forecast of quantile 0.5; the blue shaded areas show quantile forecasts of 0.2 and 0.8. Best viewed in color.

and previous time internal states to new desired outputs. This allows RNNs to capture temporal information in sequence data. Long Short-Term Memory (LSTM) [14] is an important class of RNN variants, which has additional memory-control gates and memory cells to selectively store historical information and keep long term information. RNNs and LSTMs have been widely used for learning sequential data such as signals [13, 14] and natural language sentences [8, 12].

RNNs and LSTMs have also been successfully applied to time-series data forecasting problems. Langkvist [22] reviewed recent research in unsupervised feature learning and sequence modeling with deep learning methods for various time-series data such as video frames, speech signals, and stock market prices. Bianchi [4] compared different RNN variants and showed that LSTMs outperformed others on highly non-linear sequences with sharp spikes thanks to the quick memory cell modification mechanism.

Taieb and Atiya [28] analyzed the performance of different multi-horizon forecasting strategies on synthetic datasets with different factors, such as length of time series and number of horizons. Flunkert [9] proposed a technique called *DeepAR* to make probabilistic forecasts by assuming an underlying distribution for time-series data (e.g., negative binomial distribution for counting problems). DeepAR could produce the probability density functions for target variables by estimating the distribution parameters on each time point with multi-layer perceptrons (MLPs). However, the distributional assumption is often too strong to apply to real-world datasets.

In another direction, many researches focus on generating quantile estimations for target variables by formulating forecasting problems as quantile regressions [20]. Zheng [33] proposed to minimize objective functions for quantile regressions with high dimensional predictors by gradient boosting [10]. Xu [31] proposed a quantile

autoregressive model which can output multi-horizon quantile predictions by sequentially feeding predictions of previous steps into the same NN for current prediction. Their NN-based autoregressive model is different from RNN as prediction values, instead of hidden states, are fed recursively. For the above methods, separate models have to be trained for different quantiles for multi-quantile forecasting tasks, which is inefficient in practice. Wen [29] proposed a technique called *MQ-RNN* to generate multiple quantile forecasts for multiple horizons. MQ-RNN uses an LSTM to encode the history of time series into one hidden vector, and uses two MLPs to summarize this hidden vector, together with all future inputs, into one global context feature and horizon-specific context features for all horizons. However, this global context feature is too general to capture short-term patterns.

Yu [32] proposed adding temporal dependencies of different window sizes as regularizers with the objective of minimizing forecasting errors. However, their method was designed to find temporal dependencies, such as repeating patterns in input variables, while our design aims to combine contextual information of different temporal scales and establish better latent features for forecasting. Also note that their matrix factorization method has $O(L^3)$ time complexity in which L is number of temporal dependencies (also called lag sets). It becomes impractical to utilize the entire history such as three months.

Attention mechanism was firstly introduced by Bahdanau [3] in machine translation task. They proposed an encoder-decoder structure to encode the source sentence, and apply the attention mechanism at decoding stage to decide which parts of the source sentence to attend to in order to emit the target word. The general idea of attention mechanism is to assign soft weights to a series of features to measure each feature's relevance to a given query feature, and combine them by weighted sum to obtain attended feature. Hori [17] proposed to handle multimodal data by fusing features of different modalities such as texts, audios and videos together with softly assigned weights of each modality. Cinar [6] proposed using an LSTM encoder-decoder with position-based attention model to capture patterns of pseudo-periods in sequence data. They applied the attention mechanism to explore similar local patterns in historical data for future prediction. However, it is impractical to look into the full history of time series and the selection of which part of the history to attend to relies on human knowledge. Moreover, their decoder is a feed-forward LSTM which ignores dynamic future information such as promotion and calendar events which could have strong influence in real-world time series forecasts. In addition, at each future step the hidden state depends on previous attended historical pattern representation, which could lead to error accumulation.

Different from [6], we utilize a bi-directional LSTM decoder to first propagate future information in both forward and backward directions, considering dynamic future information such as promotions and calendar events. Then at each future time step we attend to several different periods of the history and generate attention vector individually. We treat different periods of the history as different modalities and we combine them by learning relative importance of each modality for predicting current time step.

3 APPROACH

In this section, we propose our designed network architectures for multi-horizon time series forecasting. We first introduce the basic encoder-decoder structure, and then explain in detail each important component of our networks including the embedding layer, temporal attention structure, and multimodal fusion layer.

3.1 Basic encoder-decoder structure

The LSTM-based encoder-decoder model has recently been widely used in machine learning tasks [18, 27] and time series forecasting tasks [6, 9]. We adopt this sequence-to-sequence learning pipeline to encode historical (and future) input variables and decode to future predictions, but with major modifications. As shown in Figure 2, the encoder part is a two-layer LSTM which maps the history of sequences to latent representations h_{t-1} (typically hidden states at the last step) that are passed to decoder. Formally, we denote the input of each time step as x_t, the hidden state as h_t, the internal gates of the LSTM cell as i_t, f_t, o_t and c_t, The formulations of the gates, cell update and output are as follows:

$$\begin{aligned} i_t &= \sigma(W_{ix}x_t + W_{im}m_{t-1}) \\ f_t &= \sigma(W_{fx}x_t + W_{fm}m_{t-1}) \\ o_t &= \sigma(W_{ox}x_t + W_{om}m_{t-1}) \\ c_t &= f_t \cdot c_{t-1} + i_t \cdot \tanh(W_{cx}x_t + W_{cm}m_{t-1}) \\ h_t &= o_t \cdot \tanh(c_t) \end{aligned} \tag{1}$$

We abbreviate the formulation as $h_t^e = LSTM^e(x_t; h_{t-1})$, in which the superscript e indicates that they are hidden states of the encoder.

The decoder is another recurrent network which takes the encoded history as its initial state, and the future information as inputs to generate the future sequence as outputs. In our design a bi-directional LSTM (BiLSTM) is used as the decoder backbone that propagates future input features both in time order and reverse time order. This structure allows both backward (past) and forward (future) dynamic inputs to be observed at each future time step. Then the hidden states of BiLSTM are fed into a fully-connected layer or a temporal convolution layer (discussed in next subsection) to produce final predictions. We emphasize that the final prediction should be made after the information propagation in BiLSTM, in other words, we do not use prediction results of previous time steps to predict the current time step. The purpose of separating information propagation stage with prediction stage is to prevent error accumulation especially for long-horizon forecasting. Our design differs from MQ-RNN [29] by explicitly modeling temporal patterns in future with BiLSTM as designed. A bi-directional LSTM is composed of two LSTMs that propagate information in forward and backward direction respectively. Formally, we denote the hidden states emitted at time t from forward LSTM as h_t^f, from backward LSTM as h_t^b, as well as the concatenation as h_t. The formulations are defined as follows

$$\begin{aligned} h_t^f &= LSTM^f(x_t; h_{t-1}) \\ h_t^b &= LSTM^b(x_t; h_{t+1}) \\ h_t &= [h_t^f; h_t^b] \end{aligned} \tag{2}$$

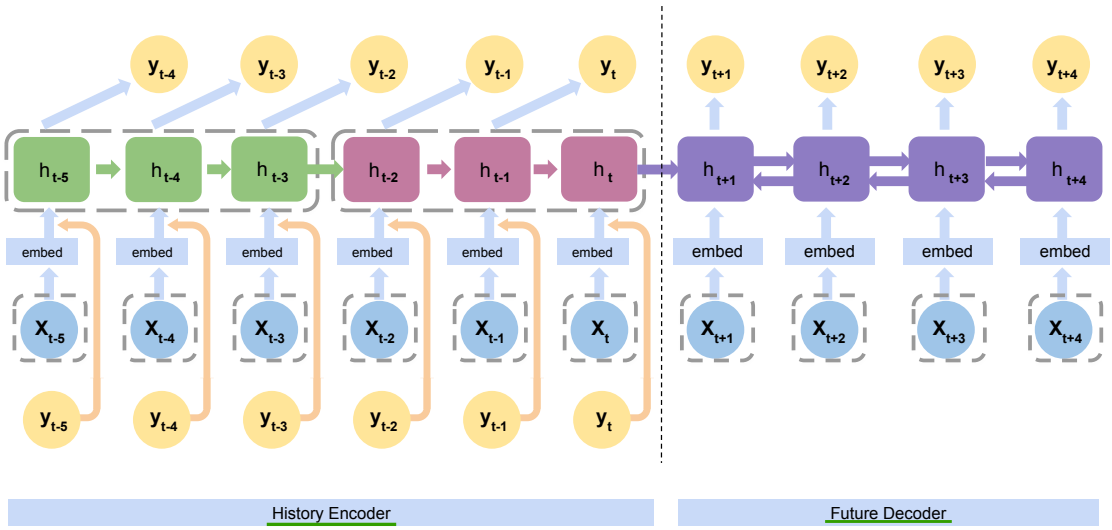

Figure 2: Structure of LSTM-based Encoder-Decoder. In the encoding stage, historical inputs x are passed through embedding layers and combined with ground truth quantities y. The output at each step is a prediction of the *next* step to regularize encoder training. In the decoding stage, future inputs are passed through the same embedding layers shared with encoder, and future quantities are predicted on top of temporal context features and evaluated with quantile losses.

We abbreviate the formulation as $h_t^d = BiLSTM^d(x_t; h_{t-1}, h_{t+1})$, in which the superscript d indicates that they are hidden states of the decoder. To generate quantile predictions from hidden states, we attach one linear layer on top of the hidden states at each time step to generate K quantile predictions at one time, as shown in the top of Fig 2. Formally, we denote K-dimensional quantile predictions at encoding and decoding stages as y_t^e and y_t^d such that

$$\begin{aligned} y_t^e &= W_e h_t^e + b_e \\ y_t^d &= W_d h_t^d + b_d \end{aligned} \tag{3}$$

We will discuss the loss functions used to evaluate the predictions and update the networks in Section 4.

3.2 Embedding

Embedding is a feature learning technique which maps categorical variables to numerical feature vectors. For example, the word embedding [24] is to learn a dictionary which maps vocabulary words to distributed and numerical feature vectors. Formally, we denote the one-hot representation of a |C|-category variable $x_c = \{0...010...0\} \in R^{|C|}$, the learnable embedding matrix as $W_c \in R^{D\times|C|}$. The transformed categorical variable can be computed as $x'_c = W_c x_c$ in which x'_c is an embedded D-dimensional feature of the input categorical variable. We concatenate multiple embedded categorical input variables, as well as the true target value in encoding stage, to form the final inputs for the model as shown in the bottom part of Fig 2.

3.3 Attention model

The encoder-decoder model shown in Fig 2 is difficult to capture long-term dependency due to its memory update mechanism which

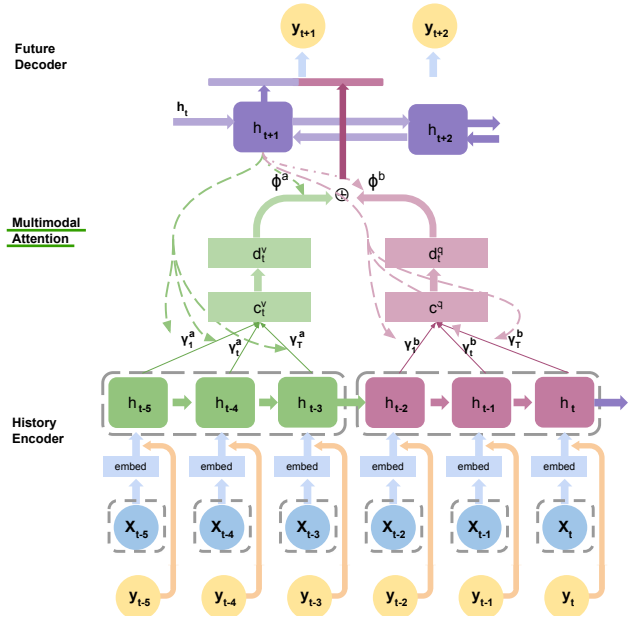

Figure 3: Structure of temporal attention modules and multimodal attention fusion layer. Embedding layers of the decoder are omitted for simplicity. Best viewed in color.

keeps erasing previous memories and refreshes with new observations. Cinar [6] proposed to use position-based attention model [2] to capture patterns of pseudo-periods in historical data. However, their model suffers from error accumulation while ignores dynamic

future information which could have significant influence on prediction accuracy. Also, it is difficult to apply their model over a long history time directly, as their attention is applied over the entire history and can be significantly diluted.

We augment with attention and describe in details in this section. In Fig 3, we illustrate our designed architecture to three parts. The bottom part is history encoder, which is identical to the history encoder shown in Fig 2. The top part is a BiLSTM future decoder which propagates available future information in both directions first, then attends to relevant history data for finding similar patterns. Our designed multimodal attention mechanism, as shown in the middle part of Fig 3, is guided by BiLSTM hidden states generated at each future time step t, and multiple attentions are applied to different parts of historical data and fused with attentional weights.

Temporal attention At decoding stage, we use BiLSTM hidden state at each future time step to attend to different parts of the history, and thus form hidden representation of future step. In contrast to [6], we do not attend to the entire history for several considerations. The first consideration is that for real-world time series data such as online sales, the history can be quite long. Attending to a long history would lead to inaccurate attention as well as inefficient computation. Instead, we propose to attend to each period of the history separately and combine them with multimodal fusion scheme which will be demonstrated in next section. The length of the period can be set to meaningful business cycle such as one month or one quarter which are typical in online sales forecasting.

Now we discuss the details of temporal attention applied on each period of the history. Following [17], the temporal attention weights $\boldsymbol{\gamma}_{1:T_h}$ are computed by

$$\begin{aligned} \mathbf{g} &= \mathbf{v}_g^\top \tanh(\mathbf{W}_g \mathbf{s}_t + \mathbf{V}_g \mathbf{h} + \mathbf{b}_g) \\ \gamma_i &= \frac{\exp(g_i)}{\sum_{j=1}^{T_h} \exp(g_j)} \qquad \text{for } i = 1 \dots T_h \end{aligned} \tag{4}$$

and shown by the dashed lines in Fig. 3. Then the attended content vectors $\mathbf{c}_t$ and the transformed $\mathbf{d}_t$ are

$$\begin{aligned} \mathbf{c}_t &= \sum_{i=1}^{T_h} \gamma_i \mathbf{h}_i \\ \mathbf{d}_t &= \mathrm{ReLU}(\mathbf{W}_d \mathbf{c}_t + \mathbf{b}_d) \end{aligned} \tag{5}$$

Multimodal fusion As shown in Fig 3, we apply temporal attention mechanism on M periods of historical data (M=2 in Fig 3), and fuse them with the multimodal attention weights $\boldsymbol{\phi}_t^{1\cdots M}$ obtained by interacting the previous hidden state $\mathbf{s}_{t\text{-}1}$ with the transformed content vectors $\mathbf{d}_t^m$

$$\begin{aligned} \mathbf{p}_t^m &= \mathbf{v}_p^\top \tanh(\mathbf{W}_p \mathbf{s}_t + \mathbf{V}_p^m \mathbf{d}_t^m + \mathbf{b}_p) \\ \phi_t^m &= \frac{\exp(p_t^m)}{\sum_{k=1}^{M} \exp(p_t^k)} \qquad \text{for } m = 1 \dots M \end{aligned} \tag{6}$$

The fused knowledge $\mathbf{x}_t$ is computed by the sum of $\mathbf{d}_t^m$ with multimodal attention weights ϕ^m such that

$$\mathbf{x}_t = \sum_{m=1}^{M} \phi_t^m \mathbf{d}_t^m \tag{7}$$

3.4 Scalability discussion

The scalability of a forecasting model is crucial as a deployed service for a large company. We now briefly discuss the scalability of our models in terms of computation resources, data size, and number of quantiles of interest. First of all, our model is scalable to computation resources. The batch size for both training and testing is linearly increasing with the number of GPUs, as most deep learning packages support parallelization over multi-GPUs in the batch dimension. Our model can be either deployed on standalone workstations or deployed over clusters. In JD.com, our sales forecasting model is deployed in multiple servers to provide a service which can be scaled with the client requests. (2) It is scalable to the size of training dataset, as the increase in dataset size will only increase the number of training iterations linearly. Moreover, as new sales data is being collected every day, our model update strategy is finetuning existing model every week in order to capture rapidly changing sales patterns influenced by trending products, seasonable changes and celebrity effects, etc. (3) It is scalable with the number of quantiles of interest. We use a linear layer to directly produce all K quantile estimations. Increase of K will linearly increase size of this layer but will have no effect on other parts of the model.

4 DATASET DESCRIPTION

We apply our approach to multi-horizon forecasting problems on a large-scale public forecasting dataset - the GEFCom2014 Electricity Price Forecasting dataset, and a huge real-world dataset with daily sales of 50,000 products on JD.com - the JD50K sales dataset. We show that our design is a unified framework which can generate accurate multi-quantile predictions on datasets of different input types and different objective functions without major changes in architectures or tricks for training.

4.1 JD50K Online Sales Forecasting

We collect a huge real-world online sales dataset from JD.com - a global online retail company. This dataset includes a total of 50,000 time series of daily sales data for different products sold in 6 regions in China from 2014 to 2018. We are interested in forecasting demand volume of each day of interested month for all products in all demand regions, and quantile predictions ranging from 0.50 to 0.95 are interested.

Sales forecasting is challenging as multiple factors have to be taken into account simultaneously such as product categories, geographical regions, promotions, etc. We briefly introduce available features as both historical and future information provided in the dataset.

- *Dc-id* indicates which distribution center (dc) of the 6 demand regions delivers a particular product. The sales difference of the same product in different regions reflects geographical preferences.
- *Cat-id* is the category index associated with each product. All 1000 products in the dataset are categorized as 96 different classes such as snacks, stationary, laptops, etc.
- *Promotion_1...4* are four binary variables indicating which promotions are in effect on each day. There are 4 different types of promotions including direct discount, threshold

discount, gift and bundle deal. Promotion events are assumed to be planned ahead and thus available as both history and future input variables.

- *Event* is the calendar variable indicating five special dates that past statistics indicated surge of sales. These are New Year's Day (January 1st), Labor Day (May 1st), National Day of China (October 1st) and two major sales festivals on June 18th and November 11th.

We evaluate forecasting algorithms with mean absolute deviation (MAD), which is defined as the sum of standard quantile loss over each time series i and each future time step t as follows

$$L^{MAD} = \sum_i \sum_q \frac{1}{T} \sum_t [q(y_{it} - \hat{y}^q_{it})^+ + (1-q)(\hat{y}^q_{it} - y_{it})^+] \quad (8)$$

where $i \in \{1, 2, ..., 50,000\}$, $q \in \{0.5, 0.6, 0.7, 0.8, 0.9, 0.95\}$.

Underestimating time series with high quantile values (> 0.5) results in higher penalties than overestimation, which is brought by the asymmetrical property of quantile loss function. Therefore, the targeted quantile q of a product can be interpreted as the "intolerance" of being out of stock, and is determined by its popularity and the targeted inventory level. We train a unified model to produce forecasts for all quantiles by forwarding once, by virtue of the linear output layer which maps the hidden context feature to all K quantiles of interest. We avoid training separate models for different quantiles because it is inefficient and unnecessary.

We sample three typical months of year 2018 as test sets, while the months before are further randomly divided to training and validation sets with ratio 5:1. The quantile of interest for each product is decided by business logic and providing the full quantile prediction is extremely beneficial for this purpose.

4.2 GEFCom2014 Electricity Price Forecasting

We also evaluate our models on the electricity price forecasting task introduced by the Global Energy Forecasting Competition 2014 (GEFCom2014) [16].

The GEFCom2014 price forecasting dataset contains three years of hourly-based electricity prices from 2011-01-01 to 2013-12-31. The task is to provide future 24-hour forecasts on 12 evenly distributed evaluation weeks. On a rolling basis, ground truth price information for previous rounds can be used as well to predict future rounds. In this dataset, hourly-based estimations of zonal and total electricity loads are two temporal features available in both past and future information. Following the competition instructions, each hourly price forecast should provide the 0.01, 0.02, ..., 0.99 quantile predictions, noted by $q_1, ..., q_{99}$. To make our settings comparable with [29], we train our models only to provide 5 quantile predictions on 0.01, 0.25, 0.5, 0.75, 0.99, while the remaining 94 quantiles are provided by linear interpolation. The quantile loss of a 24-hour forecast is defined as

$$L^{MAD} = \sum_q \frac{1}{T} \sum_t [q(y_t - \hat{y}^q_t)^+ + (1-q)(\hat{y}^q_t - y_t)^+] \quad (9)$$

in which $T = 24$, y_t and $\hat{y}_t$ are ground truth and predicted prices for t-th hour, and $q \in \{0.01, 0.25, 0.5, 0.75, 0.99\}$. To evaluate the full quantile forecasts over all evaluation weeks, losses of all target quantiles (99) for all time periods (12 weeks) over all forecast horizons are calculated and then averaged, which is an average of $12 \times 7 \times 99$ forecasts in total. A lower loss indicates a better forecast.

5 EXPERIMENTS

We evaluate our forecasting model on two large-scale forecasting datasets in different domains. We also measure the performance contribution of each component of our networks and make an ablation study for feature selection.

5.1 Training and evaluation

In this section, we describe the details of model training and evaluation. There is a total of 50,000 time series in the JD50K online-sales dataset. Each time series is split into the training and testing part. The training part starts from the beginnings of time series (as early as January 2014) and the testing part starts at the beginning of April, May or March of 2018. In addition, randomly sampled sets of consecutive days in total of one fifth of the series length per time series are held out as validation series. During training time, we randomly sample a batch of 32 different time series for each iteration. For each sampled time series, we randomly pick a training creation date, then take T_h steps past of creation date as the history and T_f steps after as the future, to form the final training sequence. Validation and testing sequences have the same length as training sequences. In real implementation, validation and testing sequences are held out in the data pre-processing step to guarantee no overlapping with training sequences. For the GEFCom2014 electricity price dataset, there is one single long series of electricity price records starting from January 2011 to December 2013. Following the settings of the competition [16], we split the time series into 12 shorter training sequences by 12 evaluation creation dates. We train 12 different models on different training sequences, and evaluate forecasting results of 99 quantiles. The average of quantile losses of 12 sequences are reported. On both datasets, we train the models up to 100 epochs while early stopping applies. The best performing models on validation sets are selected to report final performance on testing sets.

5.2 Baseline methods

We implemented several baselines to characterize the difficulty of forecasting on these datasets and confirm the effectiveness of our approach.

- *Benchmark* directly replicates historical values as future predictions. For online-sales dataset, benchmark predictions for the evaluation month are borrowed from sales quantities of previous month (31 days ago). For electricity price dataset, benchmark predictions on evaluation days are from one day before (24 hours ago).
- *Gradient-Boosting* [10] is a classical machine learning method for regression and classification problems. We use gradient boosting to learn prediction models for all future horizons, and use grid search to find optimal parameters.
- *POS-RNN* [6] is a deep learning approach that applies a position-based attention model to history of the sequence and obtains a softly-attended historical feature. It then uses a combination of historical feature and current hidden state

Methods		Quantile Loss						
		0.5	0.6	0.7	0.8	0.9	0.95	Avg
Baselines	Benchmark	5.92	5.53	5.14	4.76	4.37	4.18	4.98
	POS-RNN [6]	3.41	3.24	3.19	3.00	2.68	2.18	2.95
	MQ-RNN [29]	2.55	2.73	2.79	2.68	2.26	1.80	2.47
Our approach	BiLSTM-Enc-Dec(h=1)	2.32	2.48	2.51	2.36	1.90	1.46	2.18
	BiLSTM-Enc-Dec(h=3)	2.33	2.49	2.51	2.36	1.91	1.45	2.17
	Single-Attention(h=1)	2.31	2.47	2.48	2.32	1.85	1.39	2.14
	Multimodal-Attention(h=3)	2.28	2.43	2.45	2.29	1.85	1.39	**2.12**

Table 1: Experiment results of baselines and our models on JD50K Sales Forecasting task.

Methods		Quantile Loss					
		0.01	0.25	0.5	0.75	0.99	Avg 99
Baselines	Benchmark	3.82	3.75	3.67	3.60	3.53	3.67
	Gradient-Boosting	1.34	3.96	4.37	3.44	0.55	3.17
	POS-RNN [6]	0.22	2.76	3.82	3.86	1.90	3.05
	MQ-RNN [29]	0.18	2.48	3.48	3.43	1.37	2.68
Our approach	BiLSTM-Enc-Dec(h=1)	0.19	2.51	3.39	3.19	0.67	2.67
	BiLSTM-Enc-Dec(h=3)	0.18	2.43	3.35	3.24	0.69	2.64
	Single-Attention(h=1)	0.16	2.37	3.20	3.26	1.13	2.54
	Multimodal-Attention(h=3)	0.15	2.28	3.23	3.17	1.02	**2.48**

Table 2: Experiment results of baselines and our models on GEFCom2014 electricity price results.

from LSTM decoder for prediction at each future horizon sequentially.

- *MQ-RNN* [29] uses LSTM encoder to summarize history of the sequence to one hidden feature, and uses MLPs to make forecasts for all future horizons from the hidden feature together with future input variables.
- *TRMF* [32] incorporate temporal dependencies into matrix factorization models by adding a regularization term which measures the likelihood of observing training time series. We adopted their choices of parameters for sales prediction.

5.3 Our approach

We compared four variants of our proposed models and evaluated their performance quantitatively.

- **BiLSTM-Enc-Dec(h=1)** and **BiLSTM-Enc-Dec(h=3)** are based on the basic encoder-decoder model as shown in Fig 2. Both the LSTM encoder and BiLSTM decoder have two layers with hidden state sizes 50. We set the embedding size of each categorical variable to be 20. BiLSTM-Enc-Dec(h=1) model has single period of historical data, while BiLSTM-Enc-Dec(h=3) model has three periods. For GEF2014 data, the unit of one period of time is a week of hourly price data (24*7); while for JD50K sales data, one period of time is one month of daily sales (31 days).
- **Single-Attention(h=1)** is based on the encoder-decoder model with attention mechanism as described in Section 3.3. Similar to **BiLSTM-Enc-Dec(h=3)** , we utilize only one period of historical data, while we add temporal attention mechanism to help find correct hidden pattern for future steps.
- **Multimodal-Attention(h=3)** accepts three periods of historical data and applies multiple attentions on them individually first, and combines them with multimodal fusion, as shown in in Fig 3.

5.4 Experiment results

Table 1 and 2 summarize the experiment results for JD50K and GEFConf2014 respectively.

Results for JD50K. In Table 1, we report losses of forecasts with certain quantiles of interests (e.g., 0.5, 0.6, ..., 0.9, 0.95), as well as the total loss of all time series. We can see that the benchmark achieved a total quantile loss of 4.98 by simply replicating sales of previous month to current. POS-RNN produced a significant lower quantile loss of 2.95, by exploring historical patterns with attention mechanism. MQ-RNN performed better than POS-RNN (2.47 v.s. 2.95) as it outputs each future horizon independently and avoids error accumulation in decoding stage. Surprisingly, our basic models BiLSTM-Enc-Dec(h=1) and BiLSTM-Enc-Dec(h=3) outperformed previous methods by a good margin thanks to the use of BiLSTM decoder which propagates dynamic future information (events and promotions) in forward and backward directions. By incorporating attention mechanism, our Single-Attention(h=1) model improved BiLSTM models to 2.14, and further improved by our Multimodal-Attention(h=3) model to 2.12 which uses three months of historical data and applies attention to each month individually.

Results for GEFCom2014. Experiment results for the electricity price dataset are shown in Table 2. We report losses of five output quantiles which are the same as [29], and use linear interpolation to extend predictions to all 99 quantiles from 0.01 to 0.99 and

report the average. The benchmark method performed the worst (3.67) in terms of total of 99 quantile losses due to high dynamics in the sequence, while gradient boosting had an improved loss of 3.17 due to quantile-awareness. POS-RNN performed slightly better (3.05) than gradient boosting, but worse than the official best result of the competition 2.70 [16]. MQ-RNN achieved current state-of-the-art 2.68[1]. This hourly forecasting task has a fixed evaluation creation time (00:00 at midnight) and cyclic horizons (24 hours), which brings strong temporal dependencies in input variables. As analyzed in [29], such temporal patterns can be hard-coded into network structures as in MQ-RNN design. Reasonable predictions can be made solely based on input variables of current time point without considering contexts. This also explains why our basic BiLSTM-Enc-Dec(h=1) and BiLSTM-Enc-Dec(h=3) models did only marginally better than MQ-RNN (2.67 and 2.64 v.s. 2.68). Nevertheless, we still found that Single-Attention(h=1) , with a more powerful decoder with attention mechanism applied on one-week historical data, achieved significant better result (2.54). Finally, our Multimodal-Attention(h=3) model which considered three-week historical data by attending to them separately and combining them with multimodal fusion, achieved the lowest loss of 2.48.

Methods	MSE Loss	
	GEFCom	JD50K
Benchmark	39.98	68.46
TRMF [32]	56.01	50.32
Single-Attention(h=1)	34.86	39.15
Multimodal-Attention(h=3)	**33.78**	**38.89**

Table 3: Experiment results evaluated with MSE on GEFCom2014 and JD50K datasets.

In addition, we also compare our methods Single-Attention(h=1) and Multimodal-Attention(h=3) against Benchmark and TRMF [32] with Mean Squared Error (MSE) on both CEFCom2014 and JD50K datasets. As shown in Table 3, our proposed methods always outperformed Benchmark and TRMF by more than 20%, and Multimodal-Attention(h=3) outperformed Single-Attention(h=1) by about 3% on both datasets.

5.5 Forecasts visualization

In Figure 4, we show multi-quantile forecasts provided by Multimodal-Attention(h=3) on two evaluation weeks of distinctive patterns: the upper series have two modalities within 24 hours while the lower series have one. By observing the quantile predictions (yellow shaded areas) of 0.25 and 0.75, we show that our model is able to capture these distinct temporal patterns on future horizons by attending to the history.

In Figure 5, we show two examples of sales forecasts on two product categories on JD.com with quantile predictions of 0.2 and 0.8. Fig (a) illustrates average sales of "snack" category, while Fig (b) illustrates average sales of "seasoning" category. In each figure, magenta line shows historical sales of past three months; black line shows real price; dark blue line shows forecast of quantile 0.5; the

[1]The reported number in [29] was 2.63. Our implementation was 2.68 which may result from different implementation details.

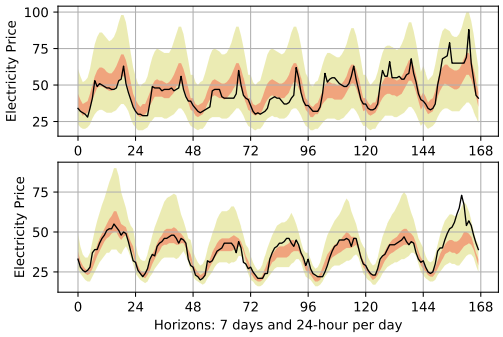

Figure 4: Electricity price forecasts of our approach on two different evaluation weeks. Black line shows real price; the lower and upper boundaries of red shaded areas show quantile forecasts of 0.25 and 0.75; the boundaries of yellow shaded areas show quantile forecasts of 0.01 and 0.99. Best viewed in color.

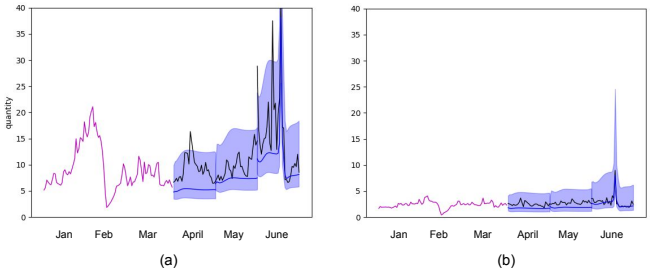

Figure 5: Examples of sales forecasts on two product categories on JD.com. Fig (a) exhibits average sales of "snack" category, while Fig (b) exhibits average sales of "seasoning" category. In each figure, magenta line shows historical sales of past three months; black line shows real price; dark blue line shows forecast of quantile 0.5; the lower and upper boundaries of blue shaded areas show quantile forecasts of 0.2 and 0.8. Best viewed in color.

lower and upper boundaries of blue shaded areas show quantile forecasts of 0.2 and 0.8. We observe that the sales of the snack shows much more drastic pattern than seasoning, while our models can capture both trends by learning from their temporal patterns.

5.6 Embedding study

We further analyze the embedding space of product category (Cat-id) to help understand what the network has learned. All category embeddings are projected to a 3-D space with t-sne [23] and we measure the similarity of two different categories by the distance in the projected space. We randomly choose four categories "Paper","Mouse","Milk" and "Roasted nuts", and list their nearest neighbors in Table 4, with cosine distances smaller than a threshold 0.25 in embedding space. We observe that most neighboring categories have associated semantic labels such as *paper and pen*, and *milk and cereal*, indicating that a proper embedding space for product category has been learned by the network. Studying category embedding space could be useful for discovering association rules

Target Category	Nearest Neighbors
Paper	Pen, Folder, Stationery, Flash Disk
Mouse	Keyboard, Flash Disk, Paper, Stationery, Router
Milk	Cereal, Grains, Dried fruit, Coffee and milk tea
Roasted nuts	Dried meat, Rice, Cereal, Beverage

Table 4: Nearest neighbors of different product categories with cosine distances smaller than 0.25**.**

based on similar sales patterns supervised by history sales. We would like to leave this topic for future study.

5.7 Implementation details

Our research code is implemented in PyTorch [26] and can be deployed in a standalone server with GPUs. During training, we updated the network parameters by Adam solver [19] with batch size 32 and fixed learning rate 10^{-3} for 100 epochs. For a typical GPU server with a single NVidia Tesla V100 GPU, we compare the inference time on JD50K dataset for forecasting one month daily sales for the total of 50,000 products. We keep the hidden size of the LSTMs the same number over all compared models for either encoders or decoders. Our Single-Attention(h=1) uses 54.1 seconds with attention over one month of historical data, and Multimodal-Attention(h=3) uses 93.7 seconds with attentions over three months of historical data. MQ-RNN [29] took a shortest time of 18.7 seconds as it has a similar LSTM encoder with ours but a much simpler MLP decoder. POS-RNN [6] took 52.6 seconds as its decoder is a feedforward LSTM, while ours is a BiLSTM which propagates future inputs in both directions and thus have more time costs.

6 CONCLUSION

In this paper, we present an end-to-end deep-learning framework for multi-horizon time series forecasting, with attention learning structures to better capture temporal contexts in historical data. We show that jointly learning temporal attentions on multiple historical periods and fusing them with multimodal attention weights is beneficial for forecasting, and our approach achieves state-of-the-art performance on two large-scale forecasting datasets.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. [n. d.]. Tensorflow: a system for large-scale machine learning.

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

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Machine Learning*.

[4] Filippo Maria Bianchi, Enrico Maiorino, Michael C Kampffmeyer, Antonello Rizzi, and Robert Jenssen. 2017. An overview and comparative analysis of recurrent neural networks for short term load forecasting. *arXiv preprint arXiv:1705.04378* (2017).

[5] G. E. P. Box and G. M. Jenkins. 1968. Some Recent Advances in Forecasting and Control. *Journal of the Royal Statistical Society* (1968).

[6] Yagmur Gizem Cinar, Hamid Mirisaee, Parantapa Goswami, Eric Gaussier, Ali Aït-Bachir, and Vadim Strijov. 2017. Position-based content attention for time series forecasting with sequence-to-sequence rnns. In *Advances in Neural Information Processing Systems*.

[7] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[8] Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science* (1990).

[9] Valentin Flunkert, David Salinas, and Jan Gasthaus. 2017. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *arXiv preprint arXiv:1704.04110*.

[10] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of Statistics* (2001).

[11] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[12] Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850* (2013).

[13] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *IEEE Conference on Acoustics, Speech and Signal Processing*.

[14] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* (1997).

[15] Charles C Holt. 2004. Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting* (2004).

[16] Tao Hong, Pierre Pinson, Shu Fan, Hamidreza Zareipour, Alberto Troccoli, and Rob J Hyndman. 2016. Probabilistic energy forecasting: Global Energy Forecasting Competition 2014 and beyond. *International Journal of Forecasting* (2016).

[17] Chiori Hori, Takaaki Hori, Teng-Yok Lee, Ziming Zhang, Bret Harsham, John R Hershey, Tim K Marks, and Kazuhiko Sumi. 2017. Attention-based multimodal fusion for video description. In *IEEE International Conference on Computer Vision*.

[18] Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[19] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

[20] Roger Koenker and Gilbert Bassett Jr. 1978. Regression quantiles. *Econometrica: journal of the Econometric Society* (1978).

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*.

[22] Martin Längkvist, Lars Karlsson, and Amy Loutfi. 2014. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters* (2014).

[23] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* (2008).

[24] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*.

[25] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499* (2016).

[26] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. (2017).

[27] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*.

[28] Souhaib Ben Taieb and Amir F Atiya. 2016. A bias and variance analysis for multistep-ahead time series forecasting. *IEEE Transactions on Neural Networks and Learning Systems* (2016).

[29] Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. 2017. A Multi-Horizon Quantile Recurrent Forecaster. *arXiv preprint arXiv:1711.11053* (2017).

[30] Peter R Winters. 1960. Forecasting sales by exponentially weighted moving averages. *Management science* (1960).

[31] Qifa Xu, Xi Liu, Cuixia Jiang, and Keming Yu. 2016. Quantile autoregression neural network model with applications to evaluating value at risk. *Applied Soft Computing* (2016).

[32] Hsiang-Fu Yu, Nikhil Rao, and Inderjit S Dhillon. 2016. Temporal regularized matrix factorization for high-dimensional time series prediction. In *Advances in Neural Information Processing Systems*.

[33] Songfeng Zheng. 2010. Boosting based conditional quantile estimation for regression and binary classification. In *Mexican International Conference on Artificial Intelligence*.