RESTFul: Resolution-Aware Forecasting of Behavioral Time Series Data

Xian Wu[1] Baoxu Shi[1] Yuxiao Dong[2] Chao Huang[1] Louis Faust[1] Nitesh V. Chawla[1]
University of Notre Dame[1] Microsoft Research[2]
{xwu9, bshi, chuang7, lfaust, nchawla}@nd.edu yuxdong@microsoft.com

ABSTRACT

Leveraging historical behavioral data (*e.g.*, sales volume and email communication) for future prediction is of fundamental importance for practical domains ranging from sales to temporal link prediction. Current forecasting approaches often use only a single time resolution (*e.g.*, daily or weekly), which truncates the range of observable temporal patterns. However, real-world behavioral time series typically exhibit patterns across multi-dimensional temporal patterns, yielding dependencies at each level. To fully exploit these underlying dynamics, this paper studies the forecasting problem for behavioral time series data with the consideration of multiple time resolutions and proposes a multi-resolution time series forecasting framework, **RES**olution-aware **T**ime series **F**orecasting (RESTFul). In particular, we first develop a recurrent framework to encode the temporal patterns at each resolution. In the fusion process, a convolutional fusion framework is proposed, which is capable of learning conclusive temporal patterns for modeling behavioral time series data to predict future time steps. Our extensive experiments demonstrate that the RESTFul model significantly outperforms the state-of-the-art time series prediction techniques on both numerical and categorical behavioral time series data.

CCS CONCEPTS

• **Mathematics of computing → Time series analysis**;

KEYWORDS

Time Series Forecasting, Multiple Resolutions, Deep Learning

ACM Reference Format:
Xian Wu[1] Baoxu Shi[1] Yuxiao Dong[2] Chao Huang[1] Louis Faust[1] Nitesh V. Chawla[1]. 2018. RESTFul: Resolution-Aware Forecasting of Behavioral Time Series Data. In *The 27th ACM International Conference on Information and Knowledge Management (CIKM '18), October 22–26, 2018, Turin, Italy*. ACM, New York, NY, USA, Article 4, 10 pages. https://doi.org/10.1145/3269206.3271794

1 INTRODUCTION

Forecasting of behavioral time series data (*e.g.*, sales volume and email communication) has benefited many real-world applications, with the aim of predicting future trends by understanding past behavioral observations [25, 33]. For example, real value regression in sales prediction and categorical label classification for email communication prediction. A breadth of research has been devoted to developing appropriate models by learning patterns in generated time series for making predictions. One of the most popular forecasting models is the Autoregressive Integrated Moving Average (ARIMA) model [32], which assumes the linear relations in behavioral time series data based on a particular statistical distribution (*e.g.*, normal distribution). Support Vector Regression (SVR) [4], another well known forecasting method, are also frequently applied in time series prediction due to the globally optimal solution they obtain. However, these approaches share a common deficiency in that they cannot capture the complex temporal dependencies, which may not properly reflect the variation in behavioral time series data across different time frequencies. To deal with the complex temporal relations, recurrent neural network models are proposed to handle non-linearities in time series data [23, 27, 39].

However, a common drawback of the above approaches is that only one time resolution (*e.g.*, daily or weekly) is considered to model the temporal patterns of behavioral time series data, which may not properly reflect the variation of time series across different time frequencies [35]. Consider retail for example, sales for a particular product are more likely to be high on a given day if that product has sold well over the previous few days. In addition, sales are impacted by weekday patterns, such as Saturdays typically being the busiest shopping days of the week. Hence, it is difficult to determine which temporal resolutions (*e.g.*, daily, weekly, monthly patterns) are more likely to yield useful patterns for making future forecasts. Because the existing research works typically use only a single resolution, temporal patterns learned by existing approaches may not be as accurate. To capture more comprehensive temporal patterns, a natural solution may lie in leveraging information from multiple time resolutions.

This presents a new problem statement: *learning temporal patterns of behavioral time series data from multiple time resolutions*. However, developing such a time series forecasting framework relies on addressing the following important technical challenges:

- **Long-range Temporal Dynamics.** Existing neural network-based time series forecasting models seldom handle the long-range dependencies in behavioral time series data due to computational efficiency, *i.e.*, time-consuming gradient derivation as it is propagated back through time steps. This oversimplification on the input sequence leads to sub-optimal forecasting results as the useful information of long-range dependency (*e.g.*, monthly or seasonal patterns) is ignored in this abstraction.
- **Lack of Comprehensive Learning**. Temporal patterns may evolve over time and as changes occur, future time series data may become more relevant to a different temporal resolution. It

CIKM'18, Oct 22-26 2018, Turin, Italy

ACM ISBN 978-1-4503-6014-2/18/10…$15.00
https://doi.org/10.1145/3269206.3271794

is challenging to summarize useful information from patterns across different time resolutions in assisting the prediction.

To tackle the above challenges, we propose a neural network framework called **RES**olution-aware **T**ime **S**eries **F**orecasting (RESTFul), that explicitly models long-range temporal dependencies in behavioral time series data. RESTFul introduces a set of "time resolution-aware sequences" to preserve time series patterns w.r.t. different time resolutions (*e.g.*, day or week). Although different resolution-aware sequences share the same length, they reflect temporal patterns over different time periods (*e.g.*, recent 5 days/months, given a sequence length of 5).

With the constructed resolution-aware sequences, RESTFul first proposes a recurrent framework to encode the evolving temporal patterns of each sequence with a low-dimensional representation. Next, RESTFul integrates the learned representations from resolution-aware sequences into a convolutional fusion framework. In essence, we aim to precisely model the complex inter-dependencies between the sequential patterns with different time resolutions. Finally, the learned conclusive embedding vector is fed into a Multilayer Perceptron (MLP) for forecasting behavioral time series data.

The advantages of the proposed RESTFul model are as follows. *First*, by considering the temporal dependencies w.r.t. different time resolutions, the proposed model outperforms state-of-the-art time series forecasting approaches in terms of prediction accuracy. *Second*, the model can automatically summarize the temporal patterns to model the behavioral data in the predicted time slots, which is obtained by considering the complex inter-dependencies between different resolution-aware time-ordered sequences. Finally, the proposed RESTFul model is robust to numerical and categorical behavioral time series data types and therefore applicable to a wide variety of real-world problems.

In summary, the main contributions of this paper are as follows:

- We present a new framework named RESTFul: Resolution-Aware Time Series Forecasting (RESTFul) to explicitly explore time-evolving temporal patterns in behavioral time series data with the consideration of different time resolutions. To the best of our knowledge, this is the first framework to generalize the behavioral time series forecasting with multiple time resolutions.
- The RESTFul framework (i) captures resolution-aware evolving dependencies in behavioral time series data with a developed recurrent framework; (ii) proposes a convolutional fusion framework to integrate temporal patterns, with respect to different time resolutions, into a conclusive representation learning architecture for making predictions.
- Extensive experiments show that our approach significantly outperforms state-of-the-art models on both numerical and categorical real-world datasets, corresponding to different types of time series forecasting applications. Specifically, our approach achieves 68.6%, 119% and 8.4% improvement over the best baseline method on Rossmann store sales, email communication and urban complaint data, respectively.

2 PROBLEM FORMULATION

In this section, we introduce necessary terminologies used in this paper and then formally present the problem statement.

Definition 1. ***Behavioral Time Series.*** *A behavioral time series is a sequential set of measurements collected at equally spaced intervals from human behavior (e.g., purchase or email communication) over a period of time. Formally, a behavioral time series can be defined as a vector* $X = [x_1, \ldots, x_t, \ldots, x_T]$ $(t \in [1, \ldots, T])$ *in a chronological order, where* x_t *is the value (i.e., continuous or discrete value) at* t*-th time interval.*

Definition 2. ***Behavioral Time Series Forecasting.*** *Behavioral time series forecasting aims to predict the future behavioral data by understanding past time-ordered observations. Formally, it can be presented as: given behavioral time series data from previous* k *time steps (i.e.,* $[x_{T-k}, \ldots, x_T]$*), the objective is to infer the value at later time steps* $x_{T'}$*, where* $T' \geq T$*.*

Given that a behavioral time series is generated in uniform intervals with different time resolutions, we further give the following time resolution definitions which are relevant to constructing the multiple resolution time series.

Definition 3. ***Interval Resolution*** α*: We define interval resolution* α *as the time frame granularity to represent the time difference between two successive data points in a time series. For example, the* α *of time series* X *is day if the constructed time series is measured on a daily basis and* α *is weekly if the constructed time series is measured on a weekly basis.*

Definition 4. ***Temporal Resolution*** β*: We define temporal resolution* β *to represent the frequency over which time period* x_t *is measured in. For example, the* β *of time series* X *is weekly if* x_t *is measured on a week-to-week frequency.*

Note that the interval resolution α should be equal to or more coarse-grained than temporal resolution β. One significant limitation of existing forecasting methods is that they only take one time series X as input, and therefore just consider a single resolution for both α and β and setting $\alpha = \beta$. In contrast, this work solves the time series forecasting problem by explicitly exploring time series patterns with different time resolutions. In particular, we model X with the combinations of different α and β with respect to different time resolution sets $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$.

3 RESTFUL: RESOLUTION-AWARE TIME SERIES FORECASTING FRAMEWORK

The RESTFul model is a two-stage representation learning framework. In first stage, we develop a recurrent framework to encode the temporal dynamics of time series data different configurations of time resolutions (*i.e.*, interval resolution α and temporal resolution β). Then, we develop a convolutional fusion framework to jointly consider the time series patterns generated from the recurrent framework, to interpret the actual time series patterns for the predicted time steps. The model illustration is shown in Figure 1.

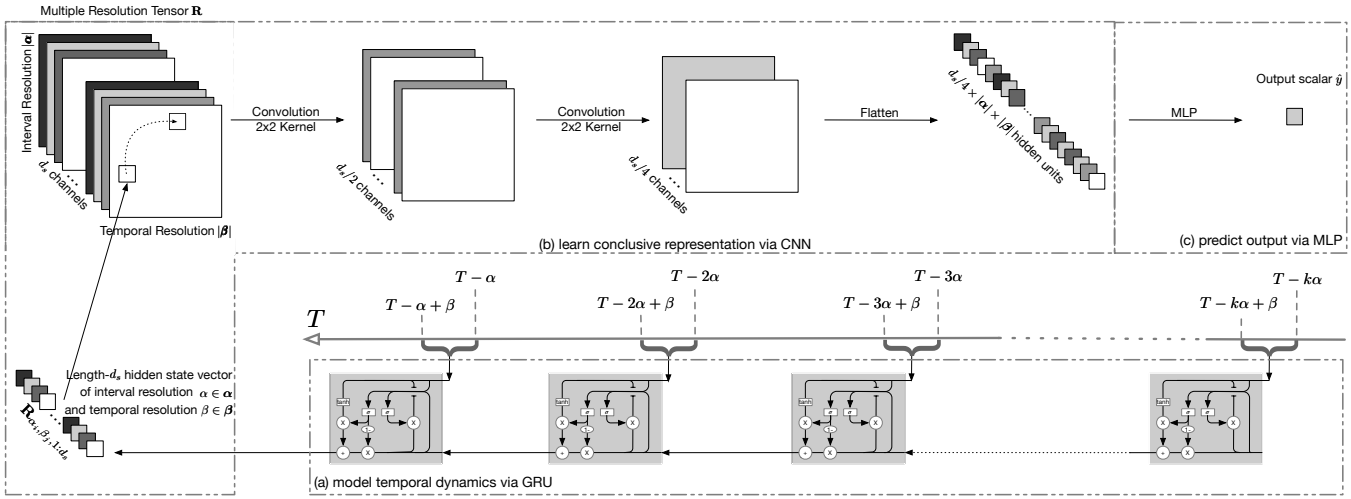


Figure 1: The RESTFul: Resolution-Aware Time Series Forecasting (RESTFul) Framework.

3.1 Modeling Temporal Dynamics via Recurrent Framework

Multi-period patterns exist in many real-world behavior time series data [1]. For example, sales of a store may spike in the first days of each season since the company introduces a new line of products at the beginning of every season, resulting in a seasonal pattern. On the contrary, sales on a regular weekday might be most relevant to sales on the same weekday last week, such as Saturdays typically being busy shopping days. Many recent works might not be able to capture these long and complicated time-dependent patterns due to their fixed resolution and limited length input sequences. Therefore, we propose to capture patterns between time series data points with different time resolutions.

Given the original time series $X = [x_1, \ldots, x_t, \ldots, x_{T-1}, x_T]$ was collected at a daily frequency, setting interval and temporal resolutions $\alpha, \beta \in \{day : 1, week : 7\}$, we can generate 3 different length-k time series as follows: (i) $\alpha = \beta = \{day : 1\}$: $X = [x_{T-k+1}, \cdots, x_{T-1}, x_T]$. (ii) $\alpha = \{week : 7\}$ and $\beta = \{day : 1\}$: $X = [x_{T-7(k+1)}, \cdots, x_{T-7}, x_T]$ where x_t is measured once a week. (iii) $\alpha = \beta = \{week : 7\}$: $X = [\overline{x}_{T-7k}, \cdots, \overline{x}_{T-14}, \overline{x}_{T-7}]$ where $\overline{x}_t = g(x_t, \cdots, x_{t+\beta})$, in which $g(\cdot)$ is some combination function (*e.g.*average operation or max operation) that generates a measurement over a period of a whole week. That is to say, we can generate $\frac{d_r(d_r+1)}{2}$ unique time series with different configurations of $\{\langle \alpha, \beta \rangle | \alpha \in \boldsymbol{\alpha}, \beta \in \boldsymbol{\beta}, \alpha \geq \beta\}$ to represent multiple periodic time series distributions, in which $d_r = \min(|\boldsymbol{\alpha}|, |\boldsymbol{\beta}|)$.

To encode the resolution-aware time series patterns, we develop a recurrent framework to capture the time-evolving dependencies in the generated time-ordered sequences with different resolutions. Recurrent Neural Network (RNN) models have been widely applicable in time series analysis. There exist various RNN architectures with different recurrent units, such as simple RNN, Long Short Term Memory (LSTM) and Gated Recurrent Units (GRU). In this section, we introduce GRU as an example for our recurrent framework for its simplicity. An illustration is shown in Figure 1.(a). The proposed recurrent framework is flexible in employing other recurrent units (*e.g.*, simple RNN or LSTM). We show how different recurrent units influence model performance in Section 4.

In particular, GRU proposes to derive the vector representations of hidden states h_t for each time step t as follows:

$$\begin{aligned} r_t &= \sigma(W_r h_{t-1} + V_r x_t + b_i) \\ z_t &= \sigma(W_z h_{t-1} + V_z x_t + b_o) \\ \widetilde{h}_t &= \phi(W_h h_{t-1} + V_h(r_t \odot \widetilde{h}_{t-1}) + b_h) \\ h_t &= z_t \odot h_{t-1} + (1 - z_t) \odot \widetilde{h}_t \end{aligned} \tag{1}$$

where $W_* \in \mathbb{R}^{d_s \times d_s}$ represents the transformation matrix from the previous state h_{t-1} to GRU cell and $V_* \in \mathbb{R}^{d_x \times d_s}$ are the transformation matrices from input to GRU cell, where d_x and d_s denotes the dimension of input vectors and hidden states, respectively. Furthermore, $b_* \in \mathbb{R}^{d_s}$ is defined as the bias term. $\sigma(\cdot)$, $\phi(\cdot)$, and $\odot$ represents the sigmoid function, tanh function, and element-wise product, respectively. In Eq. 1, r_t and z_t represent update and reset gates, respectively. For simplicity, we denote Eq. 1 as $h_t = \text{GRU}(*, h_{t-1})$ in the following subsections. The derivations of the hidden vector representation h_t for each time step t is formally given as follows:

$$\begin{aligned} h_1^{\alpha,\beta} &= \text{GRU}(x_{\alpha,\beta}^{t-k}, h_0^{\alpha,\beta}) \\ &\cdots \\ h_k^{\alpha,\beta} &= \text{GRU}(x_{\alpha,\beta}^{t-1}, h_{k-1}^{\alpha,\beta}) \end{aligned} \tag{2}$$

where $x_{\alpha,\beta}^{t-1}$ is the $(t-1)$-th observed data point in time series $X_{\alpha,\beta}$ with the corresponding interval resolution α and temporal resolution β. For each configuration of interval and temporal resolution $\langle \alpha, \beta \rangle$, we denote the corresponding hidden state vector as $h_t^{\alpha,\beta}$. In particular, given d_r different time resolutions, we can generate $\frac{d_r(d_r+1)}{2}$ sequences and thus get $\frac{d_r(d_r+1)}{2}$ hidden states.

3.2 Time Series Pattern Integration via Convolutional Fusion Framework

The key idea of convolutional fusion is to fuse the dynamic temporal patterns (with different resolutions) captured by the above recurrent framework for the final prediction. In particular, we integrate the embedding vectors learned from each individual recurrent encoder into a new conclusive embedding vector to jointly consider various time series patterns with different $\langle \alpha, \beta \rangle$ configurations.

One common way to combine multiple hidden state vectors is to directly concatenate them to generate a long vector. Then, we can capture the non-linear relations by feeding the generated vector into a Multilayer Perceptron (MLP). However, simply concatenating embedding vectors cannot capture their resolution-aware interactions as discussed in [9]. Another way to solve the data fusion problem is to employ an attention mechanism in different time series patterns. However, fusing different temporal patterns through a weighted summation fails to capture the high level insights extracted from temporal patterns in time series data [42]. In the process of time series pattern integration, while the learned sequential patterns with different time resolutions can help identify the temporal dependency in time series data, they often do not directly contribute to the final prediction due to the hierarchical interdependent relations (*e.g.*, support or mutually exclusive relationships) between temporal patterns with different time resolutions. Ignorance of such hierarchical interdependent relations will significantly impact performance as shown in our experiments.

Recently, Convolutional Neural Networks (CNN) have been applied to many natural language processing tasks to model the hierarchical relationships among words and have achieved high performance on a variety of tasks [15, 20]. Inspired by these recent advances, we propose a convolutional fusion framework to summarize a multi-resolution tensor $\mathbf{R} \in \mathbb{R}^{|\boldsymbol{\alpha}| \times |\boldsymbol{\beta}| \times d_s}$ into a conclusive representation. Here we use $\mathbf{R}$ to represent a collection of time-evolving patterns generated from multiple time resolutions. Namely, $\mathbf{R}_{i,j,1:d_s}$ denotes a learned sequence pattern representation $h_t^{\alpha_i, \beta_j}$ at time t w.r.t. temporal resolution α_i and interval resolution β_j as described in Eq. 2. We further apply a mirror padding along the diagonal to complete the tensor. The generation process of $\mathbf{R}$ is shown in Figure 1.(b).

Figure 1.(c) demonstrates the CNN fusion process. For each convolution layer in our network, we employ a filter $w \in \mathbb{R}^{l_i \times l_j \times d_s}$ to perform the convolution operation on $i \times j$ embedding vectors. In particular, this operation is performed based on the resolution position relations in terms of time interval granularity. Formally, the convolution operator is given as follows:

$$\mathbf{R}^{n+1}_{i,j,k} = f\left(w^{n+1}_k \cdot \mathbf{R}^{n}_{(i:i+l_i, j:j+l_j, 1:d_s)} + b^{n+1}_k\right) \tag{3}$$

where $b_k \in \mathbb{R}$ is a bias term and f is some activation function, n denotes the index of convolutional layer. We also apply Batch Normalization [13] and Dropout [37] after each convolution layer. Lastly, we flatten the resulting tensor to generate the final conclusive representation h.

After obtain the representation h of the current time step, we feed the reshaped one dimensional conclusive embedding vector into the Multilayer Perceptron (MLP) component and derive the targeted value (the occurrence probability for categorical value prediction). Formally, we represent MLP as follows:

$$\begin{aligned} L_1 &= \phi(W_1 h + b_1) \\ &\cdots \\ L_{d_L} &= \phi(W_{d_L} L_{d_L - 1} + b_{d_L}) \\ \hat{x}_t &= \sigma(W_x L_{d_L} + b_x) \end{aligned} \tag{4}$$

where d_L represents the number of hidden layers in MLP. ϕ, W_* and b_* represent the *ReLU* activation function, weight matrix and bias vector of each MLP layer respectively. We further apply *sigmoid* (denoted as σ) to output the predicted probability. For simplicity, we denote Equation 4 as $\hat{x}_t = \text{MLP}(h_t, d_L)$. In the experiments, we set the number of layers d_L in MLP as 2.

3.3 Learning Process

This work proposes a general framework which can be applied to a variety of prediction tasks, including regression and classification, on different types of time series (*i.e.*, numeric and categorical values). To solve different types of forecasting tasks, it is necessary to specify different loss functions accordingly. A commonly used metric for regression tasks is *mean-squared-error (MSE)* [30]. Hence we define our loss function for continuous time series prediction as:

$$\mathscr{L} = \sum_{i,t} ||x_{i,t} - \hat{x}_{i,t}||_2^2 \tag{5}$$

where $x_{i,t}$ and $\hat{x}_{i,t}$ denote the actual and estimated data value of the i-th time series in t-th time interval, respectively.

In addition, to be consistent with the our experiment, we simplify the categorical value prediction task as binary cases and employ *cross entropy* [28] in the objective function. Therefore, we define our loss function for categorical value prediction as:

$$\mathscr{L} = -\sum_{i,t} x_{i,t} \ \log(\hat{x}_{i,t}) + (1 - x_{i,t}) \ \log(1 - \hat{x}_{i,t}) \tag{6}$$

where $\hat{x}_{i,t}$ denotes the estimated probability that the data point of the i-th time series in t-th time interval. Furthermore, $x_{i,t} = 1$ if the data point of the i-th time series in t-th time interval is 1, left as 0 otherwise. The parameters of RESTFul can be achieved by minimizing the loss function in Eq. 5 and Eq. 6, corresponding to the specific type of forecasting task.

4 EVALUATION

We demonstrate the effectiveness of the proposed *RESTFul* on various numerical and categorical real-world datasets. In particular, we aim to answer the following research questions:

- **Q1**: How does our *RESTFul* model perform compared to the state-of-the-art time series forecasting models on numerical data?
- **Q2**: How does our *RESTFul* model work for time series forecasting on categorical values compared to baselines?
- **Q3:** Does *RESTFul* consistently outperform other baselines in terms of prediction accuracy on three tasks *w.r.t* different time windows with different training and testing time periods?
- **Q4**: How is the performance of *RESTFul*'s variants with different combinations in the joint framework?

- **Q5**: How do the key model hyper-parameters impact *RESTFul*'s performance?
- **Q6**: How do the number of time resolutions impact *RESTFul*'s performance?
- **Q7**: Whether the selection of recurrent unit in *RESTFul* influence the model performance?

4.1 Experimental Setup

4.1.1 **Data.** In our evaluation, we perform experiments on two types of time series data (*i.e.*, numerical and categorical data) to evaluate the performance of *RESTFul* for the aforementioned tasks: (i) sales volume prediction on Rossmann store sales data [1]; (ii) temporal link prediction on Enron email data [2]; (iii) urban complaint prediction on 311 service data [3].

(i) **Rossmann Store Sales Data**. This dataset contains 336, 808 sale records for the Rossman drug-store chain in Germany, collected from daily sales across 1, 115 stores. We utilize sales data from Jun, 2014 to Jun, 2015. The format of each sales log is: (store id, the turnover, timestamp). Hence, for each store i, we can generate a time series X_i in which each element $x_{i,t}$ is the turnover (quantitative value) of store i in t-th time step. Time step granularity is at the daily level in this dataset.

(ii) **Enron Email Data**. This email dataset was constructed by the CALO Project [22] to reflect 1, 153, 562 email communications of 150 users for understanding how an email system is used. We employ data spanning Jan, 2001 and Dec, 2001 in our evaluation. In this dataset, there exist 148, 805 user pairs and each email communication is recorded with the format of (source user id, target user id, timestamp). In particular, a link (i, j, t) is generated when user i sends an email to user j in t-th time step or vice versa. We generate a time series (*e.g.*, $X_{i,j}$) for each pair of users (*e.g.*, user i and j) to represent if they have a daily communication or not. Each element $x_{i,j,t}$ in $X_{i,j}$ is set to 1 if link (i, j, t) or (j, i, t) exists in the dataset and 0 otherwise.

(iii) **Urban Complaint Data**. This dataset is collected from 311 non-emergency services from in New York City which document people complaints of different categories (*e.g.*, noise and blocked driveway). Each reported complaint record is in the format of (complaint category, latitude, longitude, timestamp). The urban complaint dataset contains 204, 333 records and was collected from Jan to Dec, 2014. In the experiments, we first map the coordinates into 863 disjointed geographical regions according to road network information in NYC [40]. Given a particular region and complaint category, we generate a time series $X_{i,j}$ by setting each element $x_{i,j,t} = 1$ if there exists category j-th complaints reported from this region in the t-th day and $x_{i,j,t} = 0$ otherwise. In this evaluation, we focus on 4 key complaint categories (e.g., Noise, Blocked Driveway, Illegal Parking and Building Use) studied in [12].

For all the above time series datasets, the interval and temporal resolutions are set as $\{day, week, month\}$ in the evaluation.

[1] https://www.kaggle.com/c/rossmann-store-sales
[2] https://www.cs.cmu.edu/ẽnron/
[3] https://opendata.cityofnewyork.us/

4.1.2 **Reproducibility and Parameter Settings.** The hyper-parameters play important roles in the *RESTFul* framework, as they determine how the model will be trained. We summarize the parameter settings of *RESTFul* in Table 1. In our evaluation, we investigate how the different choices of parameters affect the performance of *RESTFul* by changing the tested parameter and fixing others as the default values. We implemented *RESTFul* using TensorFlow and Adam [21] to learn the parameters. We utilize the grid search method to optimize the hyperparameter settings in our experiments. In addition, we set $\alpha, \beta \in \{day, week, month\}$. The source code of *RESTFul* is publicly available [4].

Table 1: Parameter Settings

Parameter	Value	Parameter	Value
Hidden State Dimension	16	# of Time Steps	6
# of MLP Layers	2	Filter Size	2
# of Convolutional Layers	2	Dropout Ratio	0.5
BN Scale Parameter	0.99	BN Shift Parameter	0.001
Batch Size	64	Learning Rate	0.001

4.1.3 **Baselines.** In our evaluation, we consider three types of baselines: (i) conventional time series forecasting methods (*i.e.*, ARIMA and SVR); (ii) variants of Recurrent Neural Network models for time series predicitve analytics (*i.e.*, GRU, Dipole, Deep-GRU, DA-RNN); (iii) traditional neural network models (*i.e.*, MLP).

- **Support Vector Regression (SVR)** [5]: a non-parametric method for regression task based on kernel functions.
- **Auto-Regressive Integrated Moving Average (ARIMA)** [19]: It is a well-known time series prediction model for understanding and predicting future values in a time series. We also explore the seasonal ARIMA (SARIMA) by setting season length as day or week. Due to its similar performance to ARIMA, we only report the evaluation results of ARIMA.
- **Multilayer Perceptron (MLP)** [17]: It is a feed-forward artificial neural network which uses a nonlinear activation function in the learning process.
- **Gated Recurrent Neural Networks (GRU)** [7]: It is a gating mechanism in recurrent neural networks which has fewer parameters than LSTM by lacking an output gate to achieve computational efficiency. We use the hidden states generated by GRU to predict future time steps based on cross entropy.
- **Stacked Gated Recurrent Framework (Deep-GRU)** [43]: It is a stacked recurrent framework in which the output of each step is generated by a set of connected GRUs instead of a single unit. This improves the basic GRU model by increasing the model depth but still accepts a single time resolution as input. To be consistent with the number of parameters of our proposed *RESTFul*, we set the number of stacked GRU layers as 6.
- **Attention-based Bidirectional Recurrent Neural Networks (Dipole)** [27]: This method further extends the basic GRU by applying bidirectional GRUs and employs an attention mechanism to interpret the relations between the current and past values.
- **Dual-stage Attention-based Recurrent Neural Network (DA-RNN)** [34]: It is a state-of-the-art time series forecasting method

[4] https://bitbucket.org/xianwu9/restful

which consists of an encoder with an input attention mechanism and a decoder with a temporal attention mechanism.

In our experiments, we keep the number of time steps consistent for all compared algorithms. All neural network models (*e.g.*., MLP, DA-RNN, GRU, Dipole, Deep-GRU) are trained using Adam optimizer. SVR and ARIMA/SARIMA are implemented based on the statsmodels library [36]. Furthermore, based on the parameter settings reported in their individual work, we also optimize the parameter settings of all baselines using grid search strategy and their best performance are reported in the evaluation results.

4.1.4 **Evaluation Protocols.** Here, we summarize the evaluation metrics used in two types of tasks (*i.e.*, numerical value forecasting and categorical value forecasting) as follows.

- **Forecasting on Numerical Values.** To evaluate the performance of all compared methods in predicting quantitative sales volume (posed as a regression problem), we use *Mean Absolute Error (MAE)*, *Root Mean Square Error (RMSE)* and *Root Mean Square Percentage Error (RMSPE)* which have been widely used in the tasks of predicting quantitative values [10, 31]. Formally, the mathematical definitions of those metrics are presented as follows: $MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y_i}|$, $RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y_i})^2}$, $RMSPE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\frac{y_i - \hat{y_i}}{y_i})^2}$, where y_i and $\hat{y_i}$ represents the actual sales volume and estimated one, respectively. Note that MAE, RMSE and RMSPE scores are the lower the better.
- **Forecasting on Categorical Values.** To validate the performance of each method in predicting link or urban compliant existence (posed as classification problem), we use *Precision*, *Recall*, *F1-score* (trade-off between precision and recall) and *AUC* as evaluation metrics [18]. A larger Precision, Recall, F1 and AUC value signals better performance.

For time series forecasting on numerical and categorical data, we split the datasets chronologically into training (7.5 months), validation (0.5 month) and test (1 month) sets. The validation datasets are used to tune hyper-parameters and test datasets are used to evaluate the final performance of all compared algorithms. All prediction experiments are conducted across the consecutive days in the test data and the average performance is reported.

4.2 Performance Comparison on Numerical Time Series Data Forecasting (Q1 and Q3)

We now compare *RESTFul* with state-of-the-art methods in predicting the sales volume on Rossmann sales data. Table 2 shows the prediction accuracy of all compared methods from Feb to Jun. From Table 2, we have the following main observations:

(i) The proposed *RESTFul* achieves the lowest MAE, RMSE and RMPSE scores and achieves significant improvements over other methods in all cases. In particular, the average improvements of *RESTFul* over different best performed baselines range from 42.1% to 68.6% in terms of RMSPE. We believe the benefits are credited to the effective design of the conventional fusion architecture in *RESTFul*–automatically capturing the importance of time-evolving dependencies between resolution-aware time-ordered sequences.

(ii) Among the baselines, there is no obvious winner between different gated recurrent frameworks, which suggests the weak effect in modeling the temporal dynamics in numeric time series data with various gated recurrent models and the attention mechanism (*i.e.*, GRU, Dipole and DA-RNN). In contrast, the significant improvements on prediction accuracy of *RESTFul* confirms our assumption in Section 2–by providing various time series inputs with different temporal resolutions which capture multiple time-evolving dependencies from different periodic patterns, this information can improve forecasting future values. Furthermore, we observe that *RESTFul* outperforms Deep-GRU which uses the same recurrent framework with similar number of parameters, further admitting the effectiveness of modeling the multi-resolution-aware temporal patterns for time series forecasting.

(iii) We observe that *RESTFul* consistently outperforms other baselines over different time frames (*i.e.*, from Feb to Jun), suggesting that *RESTFul* is capable of handing temporal dynamics maintained by month and season variation (*e.g.*, Feb–Winter; Mar, Apr–Spring; May, Jun-Summer).

4.3 Performance Comparison on Categorical Time Series Data Forecasting (Q2 and Q3)

For the binary time series value prediction task, we report the experimental results in Table 3 and Table 4 corresponding to the temporal link prediction and complaint prediction tasks, respectively. From the evaluation results, we note the following key observations:

(i) Table 3 lists the evaluation results in predicting future email communications on the Enron dataset. We can observe that *RESTFul* performs better than competing methods in most cases in terms of F1-score and AUC. In particular, *RESTFul* outperforms its counterpart Dipole with 45.9%–119% relative improvements in terms of F1 across four different months. The performance gain between *RESTFul* and other baselines is relatively stable when varying the training and testing time windows. The advantage of the proposed framework lies in its proper consideration and accommodation of uncertain periodic influence–the existence of multiple types of dynamic time series patterns with different temporal resolutions. Additionally, we can observe that *RESTFul* consistently outperforms other baselines with respect to different density degrees of time series data, which suggests that *RESTFul* is robust to data sparsity.

(ii) We repeat the above experiments on urban complaint data to predict the future complaints of each region in a city. The evaluation results are presented in Table 4. From this table, we observe that *RESTFul* continuously outperforms all compared baselines in most evaluation metrics. For example, *RESTFul* outperforms the best baseline on November in terms of F1-score by 8.8%. In the occasional cases that *RESTFul* misses the best performance, it still achieves competitive results. The evaluation results across different time frames further validate the effectiveness of *RESTFul* in modeling time-evolving dependencies in time series data with multiple resolutions and reasonably interpret the importance of different periodic patterns in predicting future results.

(iii) We observe that *RESTFul* shows improvement over both conventional time series forecasting techniques (*i.e.*,) and variants of

Table 2: Performance comparisons of different methods in sales volume prediction in terms of *MAE, RMSE and RMPSE. RESTFul* achieves the best performance in all cases.

Month	February			March			April			May			June		
Algorithm	MAE	RMSE	RMSPE	MAE	RMSE	RMSPE	MAE	RMSE	RMSPE	MAE	RMSE	RMSPE	MAE	RMSE	RMSPE
SVR	1537	1960	0.396	1427	1861	0.382	2829	3570	0.460	2485	3045	0.375	2977	3608	0.404
ARIMA	1536	2073	0.378	1516	2055	0.287	2283	3007	0.367	1848	2481	0.320	2050	2865	0.338
ANN	1096	1496	0.401	1091	1711	0.240	1408	2030	0.338	1212	1685	0.272	1258	1928	0.259
GRU	1029	1571	0.398	1002	1597	0.243	1382	2041	0.336	1194	1604	0.275	1272	1919	0.262
Deep-GRU	1071	1568	0.405	989	1582	0.239	1356	1969	0.330	1184	1632	0.273	1233	1905	0.261
Dipole	1098	1554	0.403	1030	1658	0.234	1371	2002	0.340	1167	1626	0.277	1232	1839	0.265
DA-RNN	1066	1563	0.421	988	1554	0.238	1281	1897	0.333	1116	1557	0.271	1193	1855	0.258
RESTFul	797	1170	0.249	767	1198	0.164	1180	1629	0.199	882	1315	0.190	888	1312	0.153

recurrent neural network models (i.e., GRU, Dipole, Deep-GRU, DA-RNN). The large performance gap between *RESTFul* and conventional time series prediction approaches indicates the limitation of those methods by assuming a fixed temporal pattern of across the time series. Additionally, although the variants of recurrent neural network models are developed to capture the dynamic sequential patterns in time series data, the performance gap between *RESTFul* and those baselines stems from the fact that the time-varying distribution of future time series data might be relevant to different temporal patterns with different resolutions.

4.4 Evaluations on Variants of *RESTFul* (Q4)

We further evaluate the key components of our joint representation learning model *RESTFul* to better understand the proposed framework. We consider several variants with the aim of answering the following four questions: (i) Does the designed convolutional neural network fusion architecture play a crucial role in the joint learning model *RESTFul* to capture the temporal dynamics in time series data with different time resolutions? (ii) How is the model performance when employing an attention mechanism to jointly consider temporal patterns with different time resolutions? (iii) Is the consideration of dependencies between time resolutions helpful for making final predictions? Here we consider three variants of the proposed method to answer these questions.

- **CNN Fusion Architecture** *(RESTFul-c)*: A simplified version of *RESTFul* which directly concatenate the encoded reprentation vectors from the recurrent framework to predict the future time series data, *i.e.*, without employing mirror padding operation convolutional fusion architecture.
- **Attention Mechanism** *(RESTFul-a)*: This model utilizes the attention mechanism as the fusion framework to determine the importance of temporal patterns with different resolutions, *i.e.*, replacing the convolutional fusion framework in *RESTFul* with an attention mechanism.
- **Resolution Dependencies** *(RESTFul-r)*: This variant feeds the temporal patterns with different resolutions into the convolutional framework without considering the inherent dependencies between resolutions in terms of their time granularity (*i.e.*, day, week and month), *i.e.*, concatenating all learned embedding vectors generated from the recurrent framework in terms of their resolution granularity order.

We report the evaluation results on store sales, email communication and urban complaint data in Figure 2(a), Figure 2(b) and Figure 2(c), respectively. From these figures, we notice the full version of our developed framework *RESTFul* achieves the best performance in most evaluation metrics across various settings. In particular, we summarize three key observations:

- (i). *RESTFul* outperforms RESTFul-c (without mirror padding operation in convolutional fusion framework) in all cases, justifying the effectiveness of our developed convolutional fusion framework to concatenate the temporal patterns with different time resolutions to make final predictions.
- (ii). *RESTFul* outperforms RESTFul-a which utilizes an attention mechanism in the fusion framework, which further demonstrates the efficacy of our convolutional fusion framework in concluding the multiple resolution-aware temporal patterns via a hierarchical fusion process, rather than a simple weighted combination.
- (iii). *RESTFul* achieves better performance than the variant RESTFul-r that does not consider the inherent dependencies between time resolutions. This observation suggests the usefulness of *RESTFul* in handling the complex inter-dependencies between multiple resolution-aware time series patterns.

4.5 Parameter Sensitivity Analysis (Q5)

RESTFul involves several key parameters (*e.g.*, hidden state dimension and input sequence length in the recurrent framework, number of convolutional layers in the fusion framework). To investigate the robustness of the *RESTFul* framework, we examine how the different choices of parameters affect the performance of *RESTFul* in predicting time series data. Except for the parameter being tested, we set other parameters at the default values (see Table 1).

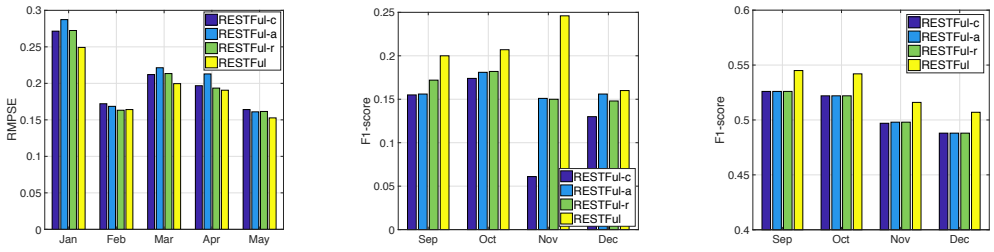
Figure 3 shows the evaluation results of urban complaint prediction (measured by F1-score) as a function of one selected parameter when fixing others. *First*, we notice that model performance becomes stable as long as the length of the input sequence is above 6. *Second*, we observe that the increase of prediction performance saturates as the dimension of the hidden state reaches 16. In our experiments, we set this dimension size as 16 due to the balance between efficacy and computational cost, *i.e.*, the smaller the embedding size is, the more efficient the training process will be. *Third*, we observe the low impact of the number of convolutional layers in the fusion framework. In summary, we observe that hyper-parameters

Table 3: Performance comparisons of different methods in temporal link prediction in terms of *AUC, F1-score (F1), Precision (Prec)* and *Recall (Rec)*. $s1$ and $s2$ represents the density degree (*i.e.*, the percentage of non-zero elements) of time series in training and testing, respectively.

Month	Sept ($s1 = 0.009$,$s2 = 0.010$)				Oct ($s1 = 0.008$,$s2 = 0.017$)				Nov ($s1 = 0.008$,$s2 = 0.013$)				Dec ($s1 = 0.009$,$s2 = 0.005$)			
Algorithm	AUC	F1	Pre	Rec	AUC	F1	Pre	Rec	AUC	F1	Pre	Rec	AUC	F1	Pre	Rec
SVR	0.307	0.019	0.401	0.010	0.356	0.021	0.547	0.011	0.391	0.036	0.374	0.019	0.313	0.031	0.287	0.016
ARIMA	0.435	0.099	0.298	0.060	0.478	0.11	0.259	0.07	0.508	0.100	0.255	0.062	0.446	0.083	0.240	0.050
MLP	0.629	0.107	0.066	0.296	0.661	0.140	0.085	0.383	0.659	0.124	0.075	0.371	0.623	0.074	0.043	0.271
GRU	0.623	0.107	0.066	0.296	0.661	0.140	0.085	0.383	0.659	0.124	0.074	0.371	0.623	0.074	0.043	0.271
Deep-GRU	0.629	0.107	0.066	0.296	0.661	0.140	0.085	0.383	0.659	0.124	0.074	0.371	0.623	0.074	0.043	0.270
Dipole	0.664	0.107	0.066	0.296	0.697	0.140	0.085	0.383	0.695	0.165	0.160	0.170	0.648	0.074	0.043	0.271
DA-RNN	0.629	0.107	0.066	0.296	0.661	0.140	0.085	0.383	0.659	0.124	0.074	0.371	0.623	0.074	0.043	0.271
RESTFul	0.833	0.200	0.193	0.207	0.792	0.259	0.315	0.219	0.629	0.246	0.232	0.261	0.810	0.160	0.178	0.146

Table 4: Performance comparisons of different methods in urban complaint prediction.

Month	Sept ($s1 = 0.139$,$s1 = 0.162$)				Oct ($s1 = 0.146$,$s2 = 0.157$)				Nov ($s1 = 0.149$,$s2 = 0.144$)				Dec ($s1 = 0.153$,$s2 = 0.137$)			
Algorithm	AUC	F1	Pre	Rec	AUC	F1	Pre	Rec	AUC	F1	Pre	Rec	AUC	F1	Pre	Rec
SVR	0.704	0.403	0.688	0.285	0.710	0.415	0.664	0.302	0.710	0.412	0.618	0.31	0.709	0.412	0.606	0.312
ARIMA	0.824	0.432	0.633	0.328	0.825	0.431	0.629	0.328	0.819	0.426	0.595	0.331	0.761	0.282	0.537	0.192
MLP	0.792	0.506	0.437	0.601	0.792	0.500	0.437	0.584	0.786	0.475	0.436	0.520	0.784	0.471	0.414	0.546
GRU	0.792	0.505	0.436	0.600	0.792	0.499	0.430	0.594	0.786	0.474	0.422	0.540	0.784	0.471	0.408	0.556
Deep-GRU	0.792	0.505	0.454	0.569	0.792	0.499	0.435	0.585	0.786	0.474	0.454	0.495	0.784	0.471	0.408	0.556
Dipole	0.792	0.506	0.443	0.589	0.792	0.499	0.425	0.605	0.786	0.474	0.415	0.551	0.784	0.471	0.414	0.546
DA-RNN	0.792	0.506	0.431	0.612	0.792	0.500	0.449	0.564	0.786	0.474	0.436	0.519	0.784	0.471	0.415	0.546
RESTFul	0.848	0.545	0.483	0.625	0.848	0.542	0.476	0.629	0.843	0.516	0.465	0.579	0.843	0.507	0.460	0.564



(a) Sales Volume (b) Email Communication (c) Urban Complaint

Figure 2: Evaluation on the variants of *RESTFul*.

have a relatively low impact on the performance of *RESTFul*, which demonstrates the robustness of the *RESTFul* framework. We also perform experiments to examine the parameter sensitivity of *RESTFul* in both sales volume prediction and urban complaint forecasting tasks where similar results were observed. Considering the space limit, we did not include them in this paper.

4.6 Effect of Resolution Configurations (Q6)

To investigate the effect of the time granularity that we used to generate time series data, we study the performance of our *RESTFul* framework with different resolution configurations, *i.e.*, (i) *RESTFul-2*: $\alpha, \beta \in \{day, week\}$; (ii) *RESTFul-3*: $\alpha, \beta \in \{day, week, month\}$. We perform experiments on three time series forecasting tasks on both numerical and categorical data. The evaluation results are shown in Table 5. In this table, we can observe that *RESTFul-3* outperforms *RESTFul-2* in most cases, which suggests that the consideration of more fine-grained time resolutions is helpful for capturing the multiple temporal patterns.

4.7 Effect of Recurrent Unit Selection (Q7)

In our experiments, we further investigate the effect of recurrent unit selection in model performance of different time series forecasting tasks. In particular, we use *(RESTFul-RNN)* and *(RESTFul-LSTM)* to represent two variants which utilizes RNN/LSTM as the basic recurrent unit to encode the time series data with temporal dynamics in the developed hierarchical model, *i.e.*, using RNN/LSTM as the recurrent unit in *RESTFul* rather than GRU. The evaluation results are presented in Figure 4. From these figures, we can observe

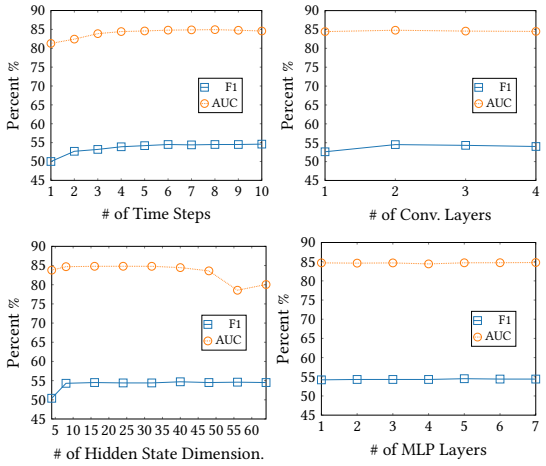
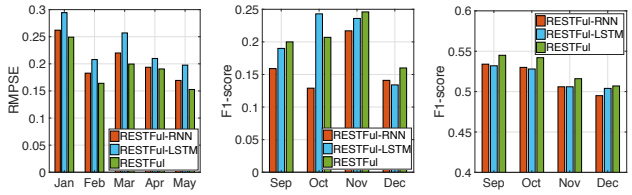


Figure 3: Parameter sensitivity of *RESTFul*.

Table 5: Performance investigation of *RESTFul* with different resolution configurations.

Data Source	Metric	RESTFul-2	RESTFul-3
Sales Volume	MAE	812	797
	RMSE	1173	1170
	RMSPE	0.251	0.249
Email Communication	AUC	0.678	0.832
	F1-score	0.206	0.207
	Precision	0.236	0.208
	Recall	0.182	0.206
Urban Complaint	AUC	0.840	0.848
	F1-score	0.537	0.545
	Precision	0.474	0.483
	Recall	0.618	0.625



(a) Sales Volume (b) Email Communication (c) Urban Complaint

Figure 4: Effect investigation of recurrent unit selection in *RESTFul*.

that *RESTFul* achieves better performance than both RESTFul-RNN (with RNN unit) and RESTFul-LSTM (with LSTM unit) in most cases. We chose GRU in our *RESTFul* framework for jointly considering both computational efficiency and prediction accuracy.

4.8 Case Study

In addition to the above quantitative analysis on sales prediction results, we further randomly select a predicted time-ordered sales sequence with ground truth information and plot the predicted numerical values of our approach *RESTFul* and other representative baselines (*i.e.*, ARIMA → representative time series forecasting technique and DA-RNN → best performed baseline from state-of-the-arts).

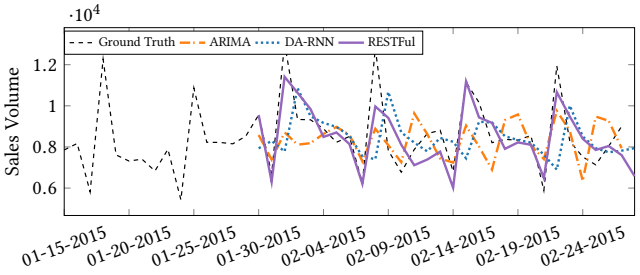


Figure 5: Case Study of Sales Forecasting.

For comparison, we visualize the prediction results of our *RESTFul* and other representative baselines over Rossmann store sales data in Figure 5. In this figure, x-axis represents the time slot index and y-axis is the sales volume. We can observe that *RESTFul* is able to capture both the sharp peak and drop of the sale sequence, while a significant delay can be observed for other baselines. This study further justifies that *RESTFul* can correctly predict the time series data with resolution-aware highly dynamic patterns.

5 RELATED WORK

Time-series Forecasting. Our work is related to recent research with a focus on modeling time series data using different techniques [8, 16, 23, 27, 29, 34, 38]. Specifically, Ma *et al.* modeled patient visit information in a time-ordered and reverse time-ordered way via attention-based bidirectional recurrent neural networks [27]. Furthermore, in [38], a hybrid framework, which integrates a convolution neural network and recurrent neural network, was developed for language sequence modeling. Laptev *et al.* proposed a LSTM-based architecture for special event forecasting at Uber. using heterogeneous time-series data [23]. This paper differs from the methods in the above work by proposing a multi-resolution time-ordered sequence prediction model that jointly explores sequential patterns from different time resolutions.

Multiple Resolution Techniques. Multiple resolution methods have been successfully used in many tasks in data mining and computer vision [2, 6, 14, 26, 45]. Specifically, Buevich *et al.* developed a multi-resolution datastore to pre-processes incoming data on embedded leaf nodes [2]. Jiang *et al.* proposed an integrated event summarization approach to facilitate the multi-resolution analysis of the events [14]. In addition, a framework has been proposed for multi-resolution spatial event forecasting [45]. Cetin *et al.* applied multi-resolution indexing for image analysis [26]. To the best of our knowledge, we are the first to address the time-series prediction problem with multiple time resolutions by employing neural network techniques.

Time-stamped Data Analytics. Our work is also related with the research work which focus on mining time-stamped data [3, 11, 24, 41, 44, 46]. For example, Cao *et al.* developed a GRU-based model with a multi-view machine layer to predict time-stamped mood scores [3]. Hu *et al.* proposed to forecast a future sub-event with a hierarchical long short-term memory architecture which encodes text information embedded in event sequences [11]. Additionally,

Zheng *et al.* developed a framework to infer air quality in a city by leveraging various urban data (*e.g.*, meteorological data and taxi trajectories) [46]. Li *et al.* aimed to identify the relationships between mood and weather data across time [24]. Note that these above methods are proposed for specific domains which incorporate various contextual features in their framework. Different from those work, our method is a general framework which lays out a solid analytical foundation to explore resolution-aware temporal patterns for forecasting behavioral time series data.

6 CONCLUSION

In this paper, we studied the behavioral time series forecasting problem with multiple time resolutions. We proposed an effective framework for allowing different resolution-aware temporal patterns to collaborate with each other and summarize a conclusive time series pattern across different resolutions. We evaluated the performance of our proposed approach on three real-world behavioral time series datasets with multiple resolutions. Experimental results on various time series forecasting tasks demonstrated the effectiveness of our RESTFul framework.

ACKNOWLEDGMENTS

This work is supported by the Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053, and NSF Grants IIS-1447795 and CNS-1622914

REFERENCES

[1] Anthony Bagnall, Jason Lines, Jon Hills, and Aaron Bostrom. 2015. Time-series classification with COTE: the collective of transformation-based ensembles. *IEEE Transactions on Knowledge and Data Engineering* 27, 9 (2015), 2522–2535.
[2] Maxim Buevich, Anne Wright, Randy Sargent, and Anthony Rowe. 2013. Respawn: A distributed multi-resolution time-series datastore. In *RTSS*. IEEE, 288–297.
[3] Bokai Cao, Lei Zheng, Chenwei Zhang, Philip S Yu, Andrea Piscitello, John Zulueta, Olu Ajilore, Kelly Ryan, and Alex D Leow. 2017. DeepMood: Modeling Mobile Phone Typing Dynamics for Mood Detection. In *KDD*. ACM, 747–755.
[4] Li-Juan Cao and Francis Eng Hock Tay. 2003. Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Transactions on Neural Networks (TNN)* 14, 6 (2003), 1506–1518.
[5] Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: a library for support vector machines. *TIST* 2, 3 (2011), 27.
[6] Dehua Cheng, Mohammad Taha Bahadori, and Yan Liu. 2014. FBLG: a simple and effective approach for temporal dependence discovery from time series data. In *KDD*. ACM, 382–391.
[7] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).
[8] Dingxiong Deng, Cyrus Shahabi, Ugur Demiryurek, Linhong Zhu, Rose Yu, and Yan Liu. 2016. Latent space model for road networks to predict time-varying traffic. In *KDD*. ACM, 1525–1534.
[9] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. World Wide Web Conferences Steering Committee, 173–182.
[10] Tracey Hollings, Andrew Robinson, Mary van Andel, Chris Jewell, and Mark Burgman. 2017. Species distribution models: A comparison of statistical approaches for livestock and disease epidemics. *PloS one* 12, 8 (2017).
[11] Linmei Hu, Juanzi Li, Liqiang Nie, Xiao-Li Li, and Chao Shao. 2017. What Happens Next? Future Subevent Prediction Using Contextual Hierarchical LSTM.. In *AAAI*. 3450–3456.
[12] Chao Huang, Xian Wu, and Dong Wang. 2016. Crowdsourcing-based urban anomaly prediction system for smart cities. In *CIKM*. ACM, 1969–1972.
[13] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*. 448–456.
[14] Yexi Jiang, Chang-Shing Perng, and Tao Li. 2014. META: Multi-resolution Framework for Event Summarization. In *SDM*. SIAM, 605–613.
[15] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188* (2014).
[16] David C Kale, Dian Gong, Zhengping Che, Yan Liu, Gerard Medioni, Randall Wetzel, and Patrick Ross. 2014. An examination of multivariate time series hashing with applications to health care. In *ICDM*. IEEE, 260–269.
[17] Dulakshi SK Karunasinghe and Shie-Yui Liong. 2006. Chaotic time series prediction with a global model: Artificial neural network. *Journal of Hydrology* 323, 1-4 (2006), 92–105.
[18] David Kempe, Jon Kleinberg, and Amit Kumar. 2002. Connectivity and inference problems for temporal networks. *J. Comput. System Sci.* 64, 4 (2002), 820–842.
[19] Mehdi Khashei, Mehdi Bijari, and Gholam Ali Raissi Ardali. 2009. Improvement of auto-regressive integrated moving average models using fuzzy logic and artificial neural networks (ANNs). *Neurocomputing* 72, 4-6 (2009), 956–967.
[20] Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* (2014).
[21] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[22] Bryan Klimt and Yiming Yang. 2004. The enron corpus: A new dataset for email classification research. *ECML* (2004), 217–226.
[23] Nikolay Laptev, Jason Yosinski, Li Erran Li, and Slawek Smyl. 2017. Time-series extreme event forecasting with neural networks at uber. In *ICML*.
[24] Jiwei Li, Xun Wang, and Eduard Hovy. 2014. What a nasty day: Exploring mood-weather relationship from twitter. In *CIKM*. ACM, 1309–1318.
[25] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. 2003. A symbolic representation of time series, with implications for streaming algorithms. In *SIGMOD Workshop*. ACM, 2–11.
[26] Vebjorn Ljosa, Arnab Bhattacharya, and Ambuj K Singh. 2006. LB-index: A multi-resolution index structure for images. In *ICDE*. IEEE, 144–144.
[27] Fenglong Ma, Radha Chitta, Jing Zhou, Quanzeng You, Tong Sun, and Jing Gao. 2017. Dipole: Diagnosis prediction in healthcare via attention-based bidirectional recurrent neural networks. In *KDD*. ACM, 1903–1911.
[28] Shie Mannor, Reuven Y Rubinstein, and Yohai Gat. 2003. The cross entropy method for fast policy search. In *ICML*. 512–519.
[29] Yasuko Matsubara and Yasushi Sakurai. 2016. Regime shifts in streams: Real-time forecasting of co-evolving time sequences. In *KDD*. ACM, 1045–1054.
[30] Gang Niu, Wittawat Jitkrittum, Bo Dai, Hirotaka Hachiya, and Masashi Sugiyama. 2013. Squared-loss mutual information regularization: A novel information-theoretic approach to semi-supervised learning. In *ICML*. 10–18.
[31] Richard J Oentaryo, Ee-Peng Lim, Jia-Wei Low, David Lo, and Michael Finegold. 2014. Predicting response in mobile advertising with hierarchical importance-aware factorization machine. In *WSDM*. ACM, 123–132.
[32] Ping-Feng Pai and Chih-Sheng Lin. 2005. A hybrid ARIMA and support vector machines model in stock price forecasting. *Omega* 33, 6 (2005), 497–505.
[33] Spiros Papadimitriou, Jimeng Sun, and Christos Faloutsos. 2005. Streaming pattern discovery in multiple time-series. In *VLDB*. VLDB Endowment, 697–708.
[34] Yao Qin, Dongjin Song, Haifeng Cheng, Wei Cheng, Guofei Jiang, and Garrison Cottrell. 2017. A dual-stage attention-based recurrent neural network for time series prediction. *IJCAI* (2017).
[35] Chotirat Ann Ralanamahatana, Jessica Lin, Dimitrios Gunopulos, Eamonn Keogh, Michail Vlachos, and Gautam Das. 2005. Mining time series data. In *Data mining and knowledge discovery handbook*. Springer, 1069–1103.
[36] Skipper Seabold and Josef Perktold. 2010. Statsmodels: Econometric and statistical modeling with python. In *Python in Science Conference*, Vol. 57. SciPy society Austin, 61.
[37] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *JMLR* 15, 1 (2014), 1929–1958.
[38] Chenglong Wang, Feijun Jiang, and Hongxia Yang. 2017. A Hybrid Framework for Text Modeling with Convolutional RNN. In *KDD*. ACM, 2061–2069.
[39] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. 2017. Recurrent recommender networks. In *WSDM*. ACM, 495–503.
[40] Xian Wu, Yuxiao Dong, Chao Huang, Jian Xu, Dong Wang, and Nitesh V Chawla. 2017. UAPD: Predicting Urban Anomalies from Spatial-Temporal Data. In *ECML/PKDD*. Springer, 622–638.
[41] Xian Wu, Yuxiao Dong, Baoxu Shi, Ananthram Swami, and Nitesh V Chawla. 2018. Who will Attend This Event Together? Event Attendance Prediction via Deep LSTM Networks. In *SDM*. SIAM, 180–188.
[42] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*. 2048–2057.
[43] Rose Yu, Yaguang Li, Cyrus Shahabi, Ugur Demiryurek, and Yan Liu. 2017. Deep learning: A generic approach for extreme condition traffic forecasting. In *SDM*. SIAM, 777–785.
[44] Xuchao Zhang, Liang Zhao, Arnold P Boedihardjo, Chang-Tien Lu, and Naren Ramakrishnan. 2017. Spatiotemporal Event Forecasting from Incomplete Hyper-local Price Data. In *CIKM*. ACM, 507–516.
[45] Liang Zhao, Feng Chen, Chang-Tien Lu, and Naren Ramakrishnan. 2016. Multi-resolution Spatial Event Forecasting in Social Media. In *ICDM*. IEEE, 689–698.
[46] Yu Zheng, Furui Liu, and Hsun-Ping Hsieh. 2013. U-air: When urban air quality inference meets big data. In *KDD*. ACM, 1436–1444.