Learning to Predict Bus Arrival Time From Heterogeneous Measurements via Recurrent Neural Network

Junbiao Pang, Jing Huang, Yong Du, Haitao Yu, Qingming Huang, *Fellow, IEEE*, and Baocai Yin

***Abstract*— Bus arrival time prediction intends to improve the level of the services provided by transportation agencies. Intuitively, many stochastic factors affect the predictability of the arrival time, *e.g.*, weather and local events. Moreover, the arrival time prediction for a current station is closely correlated with that of multiple passed stations. Motivated by the observations above, this paper proposes to exploit the long-range dependencies among the multiple time steps for bus arrival prediction via recurrent neural network (RNN). Concretely, RNN with long short-term memory block is used to "correct" the prediction for a station by the correlated multiple passed stations. During the correlation among multiple stations, one-hot coding is introduced to fuse heterogeneous information into a unified vector space. Therefore, the proposed framework leverages the dynamic measurements (*i.e.*, historical trajectory data) and the static observations (*i.e.*, statistics of the infrastructure) for bus arrival time prediction. In order to fairly compare with the state-of-the-art methods, to the best of our knowledge, we have released the largest data set for this task. The experimental results demonstrate the superior performances of our approach on this data set.**

***Index Terms*— Bus arriving time prediction, recurrent neural network, heterogenous measurement, long-range dependencies, multi-step-ahead prediction.**

Manuscript received February 2, 2017; revised July 8, 2017, December 17, 2017, April 27, 2018, and July 25, 2018; accepted September 29, 2018. Date of publication October 31, 2018; date of current version August 27, 2019. This work was supported in part by the Natural Science Foundation of China under Grant 61672069, Grant 61872333, and Grant 61620106009, in part by the China Post-Doctoral Research Foundation, in part by the Beijing Municipal Commission of Education under Grant KM201610005034, and in part by the Beijing Municipal Commission of Transport Science and Technology Project. The Associate Editor for this paper was E. Herrera-Viedma. *(Corresponding authors: Junbiao Pang; Haitao Yu.)*

J. Pang is with the Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China (e-mail: junbiao_pang@bjut.edu.cn).

J. Huang is with IBM China Investment Company Ltd., Beijing 10085, China (e-mail: huangjing_93@126.com).

Y. Du and H. Yu are with the Beijing Transportation Information Center, Beijing 100161, China (e-mail: duyong@bjjtw.gov.cn; yuhaitao@bjjtw.gov.cn)

Q. Huang is with the School of Computer and Control Engineering, University of Chinese Academy of Sciences, Beijing 100049, China, and also with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China (e-mail: qmhuang@ucas.ac.cn).

B. Yin is with the Faculty of Electronic Information and Electrical Engineering, Dalian University of Technology, Dalian 116024, China, and also with the Beijing University of Technology, Beijing 100124, China (e-mail: ybc@dlut.edu.cn).

This paper has supplementary downloadable multimedia material available at http://ieeexplore.org provided by the authors. This supplementary file provides the settings for the baseline methods. The total size of the file is 0.19 MB.

Digital Object Identifier 10.1109/TITS.2018.2873747

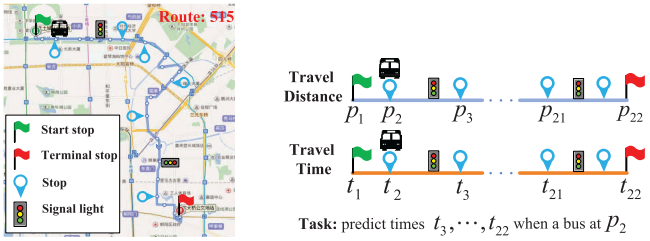
I. Introduction

TRAFFIC plays an important role in the modern urban society. Based on the report from United Nations Population Division, almost 64 billion people will live in urban areas in 2050, and the growth rate of urban population is increasing around 1.5% [1] annually. It is expected that a continuous increase of travel demand will occur in cities. Owing to the limitation of the traffic resources, a shared passenger transportation service is possible to relieve the total traffic pressure [2]; besides, the environment pressure is eased by the extensive applications of public transport in terms of the energy consumption [3].

Considering the benefits above, public transport authority is attempting to not only provide the adequate public transportation service but also enhance quality of service. Accurate bus arrival time prediction is crucial to this purpose [4]–[6]. For instance, arrival time prediction facilitates the design, development and management of the Bus Rapid Transit (BRT) system and the bus arrival notification systems of the Advanced Public Transportation System (APTS). Owing to its importance, bus arrival time prediction has always been an active research topic over several decades, dating back at least to 1966 [7].

Bus arrival time is difficult to be predicted in the real world, due to the stochastic nature of transportation, *i.e.*, delay at an intersection, the fluctuation of the travel demand over space and time, and the influence caused by weather. Most methods about bus arrival prediction have been discussed in [8]–[10]. Essentially the design of a prediction system involves two major aspects: (1) features extracted from heterogenous measurements; and (2) an efficient prediction model. Currently, Global Position System (GPS) has been solo used for bus arrival time prediction [11]. However, as O'Sullivan *et al*. [12] discussed from the perspective of the uncertainty of arrival time prediction: "The prediction problem is complicated as bus travel times are the result of nonlinear and complex interactions of many different constituent factors influencing either demand or capacity." On the other hand, frequently used prediction approach [13]–[15] predicts the arrival time for the *next* stop iteratively, which only correlates two consecutive stops. Hereafter, we name this approach as One-time-Step Ahead Prediction (OSAP). For instance, kernel machines [16] and neural network [17] have been proposed to predict arrival time.

Although bus arrival time is intrinsically characterized by heterogenous factors (the dynamic factors, *e.g.*, on-line GPS



(a) (b)

Fig. 1. The arrival time prediction is a multi-step-ahead task (best viewed in color). (a) A bus route is influenced by many static factors. (b) Multi-step-ahead prediction.

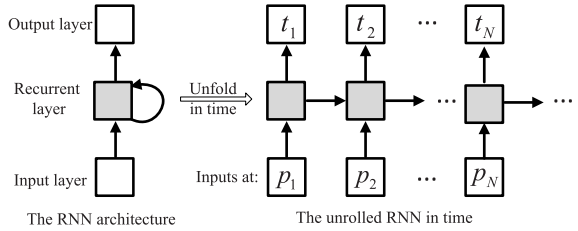


Fig. 2. The RNN architecture is unrolled into a full network along time which describes a complete time sequence.

data, and the static ones, *e.g.*, intersections), many prediction approaches just leverage GPS data. On the other hand, most of the prediction algorithms are not aware of the essence of bus arrival time prediction – it is a Multi-Step-Ahead Prediction (MSAP) [12]. Take an example in Fig. 1(b), the arrival time prediction for the p_{21} stop should exploit the correlations among the *multiple* stops (*i.e.*, $p_3, \ldots, p_{20}$) ahead; however, the corresponding arrival times $t_3, \ldots, t_{20}$ would not be observed but have to be estimated.

Therefore, it is natural to ask whether the heterogeneous measurements can be efficiently exploited, at the same time, the long-range dependencies among multiple steps are efficiently leveraged. There are many potential benefits behind this idea: reducing the uncertainty of the prediction [12], improving the prediction performances, and increasing the robustness of a prediction system. Besides, we want to avoid the disadvantages of the popular OSAP-based methods, such as Kalman filter [15], [18]: the intrinsic linearity, and the memoryless property. In summary, our goal is to propose a *nonlinear* prediction model which exploits the *heterogeneous* measurements and *long-range* dependencies across multiple time steps.

Both the long-range dependencies and the non-linearity power of Recurrent Neural Network (RNN) [19] are theoretically attractive to bus arrival time prediction. RNN is a class of neural network where connections between units in the hidden layer form a directed cycle. Fig. 2 shows a simple example, explaining the basic idea behind RNN: the unrolled RNN generates a time sequence based on the previous observations.

However, training RNN is difficult due to either exploding gradient or vanishing gradient [19] which eventually deteriorates the long-range dependencies into the short ones [20]. Therefore, the gradient-stabilizing architectures, such as Long Short-Term Memory (LSTM) [20] and Gated Recurrent Unit (GRU) [21], are proposed. RNN with LSTM (or GRU) block is firmly evidenced to build the long-range dependencies (*e.g.*, excess of 1,000 time steps), without loss of the lag capabilities [22] in the short time. Therefore, MSAP can be handled by generating the multiple ahead estimations via RNN with LSTM.

To the best of our knowledge, this paper is the first to investigate MSAP for bus arrival time prediction, presenting a comprehensive series of the experiments to illustrate the benefits of this novel idea. The proposed method is conceptually simple, yet exceptionally powerful. Simply by correlating multiple heterogeneous measurements into the long-range dependencies, with no further parameter tuning for different routes, we find a bus arrival time predictor that meets or exceeds the current state-of-the-art methods! Our main contributions are summarized as follows:

- To the best of our knowledge, we release the largest data set consisting of the GPS data from buses and the static information from a road network, establishing a new benchmark data set for this task.[1] This is very important for the development of the bus arrival time prediction task. Because without an unbiased and publicly-available data set, we neither repeat the published results nor know which idea is more efficient than the others.
- We introduce RNN with the LSTM block to build the long-range dependencies for bus arrival time prediction. This is the first to address the relation between MSAP and the long-range dependencies.
- One-hot coding is introduced to encode heterogeneous measurements into a vector space. This scheme is crucial to encode multiple measurements into a learning system.

The rest of this paper is organized as follows: Section II reviews the related work. We describe the details of the largest data set in Section III, and then elaborate on the proposed method in Section IV. Experimental results are presented in Section V and the paper is concluded in Section VI.

II. Related Work

We only present an overview of the representative methods here. For an exhaustive reference, please refer to [8]–[10]. Technically, bus arrival time prediction can be divided into three categories: the regression-based, the filter-based, and the searching-based methods.

A. The Regression-Based Method

It is reasonable to assume that the bus arrival time is an output of a function of the spatio-temporal variables. Therefore, the difficulty of this approach is how to define the non-linear relationship between the arrival time and spatio-temporal factors.

Both Support Vector Machines (SVM) and Supprot Vector Regression (SVR) have demonstrated their success in time-series analysis [8], [16], [23]. For instance, Yu *et al.* used SVM to predict bus arrival time by considering the segment-level travel time and weather conditions. However,

[1]This data is for any non-commercial research. For any interested one, please contact the corresponding authors.

the non-linearity of SVM and SVR comes from the kernel trick which is not scalable for a large-scale problem [24].

Neural Networks (NNs) have gained popularity due to their non-linearly modeling ability. Ramakrishna *et al.* [25] demonstrated the superior performances of Multiple Layer Perceptron (MLP) (a special forward NN [26]) over the linear regression approach. Chien *et al.* [17] put the historical arrival time, the travel speed, and the dwell time into MLP in a simulated scenario. However, [17] and [25] use a kind of a forward network which fails in modeling MSAP. In contrast, the basic component of our approach is a *recurrent* network. Technically, RNN used in our approach recurrently builds the long-range dependencies, guaranteeing a better performance than that of MLP.

Recently, State Space Neural Network (SSNN) has been used for the travel time prediction task [27]–[29]. For example, Lint *et al.* [27] handled the corrupted input in SSNN for the freeway travel time prediction. Theoretically, SSNN is a special case of RNN, only modeling the short-dependencies between two connective time steps [30]. Therefore, the main difference between the SSNN-based method [27] and the proposed method is that whether the long-range dependencies have been established or not. Empirical results in Section V-D indicated that long-range dependencies are important to guarantee the performance for bus arrival time prediction.

B. The Filter-Based Method

Following the classical online Bayesian prediction, the filter based method assumes that arrival time is predicted based on the last one road segment.

Kalman Filter (KF) has been widely applied to this task [18]. For instance, Vanajakshi *et al.* [15] assumed that the travel time is influenced by the mixed traffic flow. Chien and Kuchipudi [13] considered the effect of the signal timing at intersections. The KF-based method has difficulty to build reliable dynamics of buses on our data set. Because these methods need a probe bus to estimate the dynamic term in KF.

C. The Searching-Based Method

To explicitly avoid handling the stochastic patterns in traffic, the "predicting-by-replacing" strategy is proposed. Concretely, averaged historical data is directly used for prediction.

k-Nearest Neighbor (k-NN) is one of the most popular approaches [11]. For instance, Liu *et al.* [31] modified the k-NN by combining clustering analysis with principal component analysis. The methods [32], [33] compare historical trips with the trace of a running bus. k-NN however is computationally intensive if a large-scale number of historical trips are collected.

III. Data Set and Preparation

A. Data Description

The data set consists of two types of data: (1) the dynamic information about the locations of buses (*i.e.*, GPS data from buses) and (2) the static information about the road network (*i.e.*, the locations of bus stops, the locations of intersections, and the statistic of each bus stop).

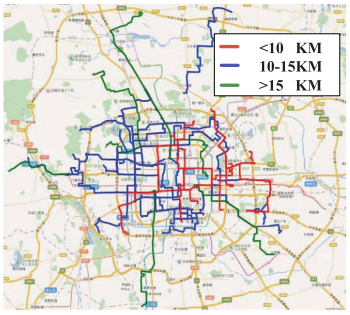


Fig. 3. The spatial distribution of 47 routes used to build the dataset (best viewed in color).

Concretely, as illustrated in Fig. 3, the GPS data are collected from 47 routes consisting of 1,089 buses in Beijing, China, during the period from Feb., 1, 2015 to Feb., 28, 2015. Each GPS datum contains the following information: the position of a bus (longitude, latitude), the time stamp, the bus ID, and the route ID. The average frequency of the GPS data is 1 point per 30 seconds, and its location error is no more than 50 meters.

The static information includes the location of a stop, the route ID, the travel directions, the locations of intersections, and the number of routes marked by a bus stop. In fact, more information, such as, automatic passenger counters, automated fare collection [35], can be easily incorporated into our method (as will be discussed in Section IV).

B. Organizing GPS Data Into Trips

Both the GPS data and the static information are mapped into the arrival time - distance data pairs. By calculating the distance between two consecutive GPS data, the travel distance for each trip is obtained. Next, we should handle the following problems: (1) projecting each bus stop onto the travel distance, and (2) obtaining the arrival time for each stop.

To handle the mentioned problems, we mapped the GPS coordinate of each stop onto the travel distance, and obtained the arrival times by an interpolation technology. In this work, Gaussian Process (GP) [36], a generalization of Kriging interpolation [37], was used for this purpose. Finally, the GPS data and the locations of bus stops are mapped into a spatio-temporal trace. A trace is termed as a trip in this paper.

Definition 1 (A Trip): Let vector $\mathbf{d} = [d_1, \ldots, d_N]^T$ denote the travel distances of N points (*i.e.*, GPS data and the GPS coordinates of bus stops) for a route, and let vector $\mathbf{t} = [t_1, \ldots, t_N]^T$ be the corresponding arrival time. A trip is a tuple consisting of the travel distance and the corresponding arrival time for each point, *i.e.*, $trip = (\mathbf{d}, \mathbf{t})$.[2]

In this paper, 22 days (with 64,245 trips) and 1 day (with 3,464 ones) are used for training and testing, respectively. The data set is summarized in Table I. As listed in Table I, the number of routes in our data set is the largest one. The lengths of bus routes range from 5.42 KiloMeter (KM) to 24.58 KM. Besides, about 95% routes in our data set are

[2]In the following, we ignore or abuse the superscript and the subscript in the different context, if it does not cause any confusion.

TABLE I
COMPARISONS AMONG DIFFERENT DATASETS

Dataset	#Route	Length(KM) of a route		#Stop/route		#Bus/route		#Trip/route	
		Min/Max	Mean± Std	Min/Max	Mean± Std	Min/Max	Mean± Std	Min/Max	Mean± Std
[32]	1	18.87/18.87	**18.87**	**59**/59	59	N/A*	N/A	1570/1570	1570
[34]	4	4/15	11±5.23	15/54	**27.75±17.90**	N/A	N/A	1276/**7882**	**3324.50±3082.79**
[8]	2	0.62/0.72**	0.67±0.07	3/3	3	N/A	N/A	**224**/237	229.67±6.66
Ours	**47**	**5.42**/**24.58**	12.02±4.29	**6**/39	18.08±6.28	**11/58**	**22.67±8.81**	718/3009	1606.53±527.62

*N/A means the data are not available.
** Note that [8] selected only two road segments(partitioned by 3 stops) for a case study. Therefore, their length of a route is relative small.

within the 4-th ring round road in Beijing, where most human activities occur in this city. All above factors make this data set very challenging for the bus arrival time prediction task.

In summary, we attempt to release an unbiased, fair and challenging data set for bus arrival time prediction. Moreover, we do hope that other researchers will test their methods on this public data set, so as to contribute to the bus arrival time prediction task.

IV. METHODOLOGY

A. The Formulation of Problem

In this subsection, we firstly present the framework of the bus arrival time prediction, and then discover why the long-range dependencies are used for this task.

Definition 2 (Framework of Bus Arrival Time Prediction): Let $trips$ denote a set of trips of a route. Let $network$ be the road network. Let $static$ be the static information of a route and let $other$ be the other possible cues. If a bus is arriving at the bus stop p_k with the travel time t_k, $(k = 1, \ldots, K)$, the framework is to learn a function as follows:

$$\hat{t}_{k,k+\Delta} = f(trip, network, static, other, p_k, t_k), \qquad (1)$$

where $\hat{t}_{k,k+\Delta}$ is the predicted arrival time at the k-th stop for a set of ahead ones $\Delta = \{1, \ldots, K-k\}$.

Specially, if $\Delta = 1$, the arrival time can be accurately predicted with off-the-shelf algorithms, such as, KF [32], [33]. Because two consecutive time steps are closely correlated. However, Δ is often larger than 1 in the bus arrival time prediction. Therefore, these approaches [38] designed for OSAP are not suitable for the MSAP task.

A common solution is to bootstrap along time steps – use the predicted $\hat{t}_k$ as the input of the $k+1$ time step. This scheme faces the problem that errors would be propagated into further time steps without the "correctness" steps [38]. This is the reason why these high efficient algorithms for the OSAP task achieve unsatisfied results in the arrival time prediction task.

An amendment is to exploit the long-range dependencies among the multiple time steps, with the aim to correct a prediction system. Concretely, it is desirable to build a correlation method to correct the predicated arrival times for the stops $p_{k+\Delta}$.

B. RNN With LSTM for Prediction

1) Measurement Selection and Representation: Based on the above discussion, it is necessary to design multiple measurements for prediction. These measurements can be organized into three categories:

1. **Spatial measurements:** the location of the passed stop d_k, and the position of the ahead stop $d_{k+\Delta}$;
2. **Temporal measurements:** the arrival time t_k, the departure time of a bus at the start stop t_0;
3. **Static measurements:** the number of routes marked by a bus stop r_k.

These variables are padded into a vector to describe the spatio-temporal information of the stop p_k:

$$\mathbf{x}_k = [t_0, t_k, r_k, d_k, d_{k+1}]^T, \qquad (2)$$

where d_{k+1} tells an algorithm where the next bus stop is.

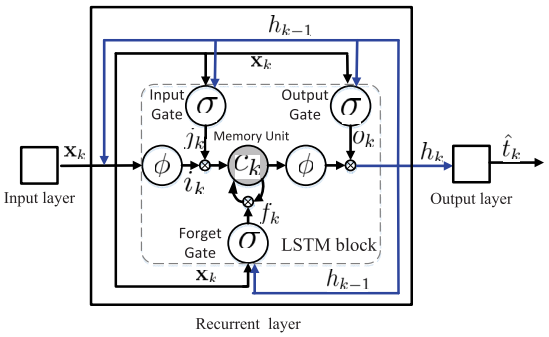


Fig. 4. LSTM RNN architecture (best viewed in color).

2) Learning to Predict by LSTM Architecture: The LSTM contains special units called *memory blocks* in the recurrent hidden layer. The memory block contains the special multiplicative units called gates to control the flow of information. Concretely, each memory block contains an *input gate*, an *output gate* and a *forget gate*. The input gate controls the flow of the input activations into the memory cell. The output gate controls the output flow of the cell into the rest of the network. The forget gate prevents LSTM to continually process the input streams. Fig. 4 illustrates both the data flow and the information flow of LSTM.

Let the sequence $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_{K-1})$ and the prediction time $\mathbf{T} = (t_2, \ldots, t_K)$ be sampled from a training set $\mathcal{X} = \{(\mathbf{X}_i, \mathbf{T}_i), \ldots\}$. An LSTM network computes a mapping from an input sequence $\mathbf{X}_i$ to an output one $\mathbf{T}_i$ via the network unit activation iteratively from $k = 1$ to $K - 1$:

$$\text{Input:} \quad i_k = \phi\left(W_{ix}\mathbf{x}_k + W_{ih}h_{k-1} + b_i\right) \qquad (3)$$

$$\text{Input Gate:} \quad j_k = \sigma\left(W_{jx}\mathbf{x}_k + W_{jh}h_{k-1} + b_j\right) \qquad (4)$$

$$\text{Forget Gate:}\quad f_k = \sigma\left(W_{fx}\mathbf{x}_k + W_{fh}h_{k-1} + b_f\right) \tag{5}$$

$$\text{Hidden State:}\quad c_k = i_k \odot j_k + f_k \odot c_{k-1} \tag{6}$$

$$\text{Output Gate:}\quad o_k = \sigma\left(W_{ox}\mathbf{x}_k + W_{oh}h_{k-1} + b_o\right) \tag{7}$$

$$\text{Block Output:}\quad h_k = o_k \odot \phi(c_k) \tag{8}$$

$$\text{Output layer:}\quad \hat{t}_k = W_{th}h_k + b_t \tag{9}$$

where W_* are the weight matrices, and b_* are the biases. The operation $\odot$ denotes the element-wise vector product. ϕ and σ are $\phi(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$, and $\sigma(x) = \frac{1}{1+\exp(-x)}$, respectively. Note that the matrices W_* and the biases b_* are shared across all time steps. In this paper, we name the prediction model built on LSTM as LSTM RNN.

The hidden state of LSTM is the concatenation (h_k, c_k). Therefore, LSTM has two kinds of hidden states: a "slow" state c_k that builds the long-range dependencies, and a "fast" state h_k which makes complex decisions over a short period.

More specially, rather than computing c_k from c_{k-1} directly with a matrix-vector product as in the vanilla RNN, LSTM computes $\Delta_k = i_k \odot j_k$, in which Δ_k is then added to update c_k. This immediately implies that the long-range dependencies are built.

Note that the length of a sequence is not equal to the maximum length of the dependencies. Because the forget gate (5) of LSTM at each point *adaptively* decides whether the accumulated history information is ignored or not. As a result, for the k-th point in a sequence, the maximum length of the dependencies is no more than $k-1$; besides, in order to make a prediction, the time complexity of the k-th point is $O(k-1)$ due to the addition operation in (6).

Theoretically, increasing the length of the dependencies would improve the accuracy of a system, if the other components of a system is properly designed and implemented. However, in practice, the number of dependencies required for a point is determined by the physical property of each stop. Fortunately, the gate mechanism empowers LSTM to *adaptively* learn the dependencies, elegantly avoiding this problem. In Summary, both the adaptiveness and the long-range dependencies of LSTM make RNN learn the complex spatio-temporal patterns.

We would like to mention Model Predictive Control (MPC) [39], which is seemingly similar to the proposed MSAP at the first glance. However, they are different in motives and techniques: 1) MPC selects the inputs to reach a predefined target in a limited number of time steps; MSAP is a special method to predict the unknown arrival time. 2) MPC optimizes a N-move control sequence that makes the output be gradually equal to the target; in order to predict the arrival time for the k-th stop, MSAP predicts the arrival times of $k-1$ stops. 3) Without the training process, MPC is driven by a predefined target; MSAP is learned from the history training data.

3) Loss Function: We simply train a LSTM RNN for each route. Therefore, the loss L is computed by a set of time sequences as follows:

$$L\left(t_{k+\Delta}, \hat{t}_{k+\Delta}\right) = \sum_{k=1}^{K-1}\sum_{\Delta=1}^{K-k} \text{smooth}_{\ell_1}\left(t_{k+\Delta} - \hat{t}_{k+\Delta}\right), \tag{10}$$

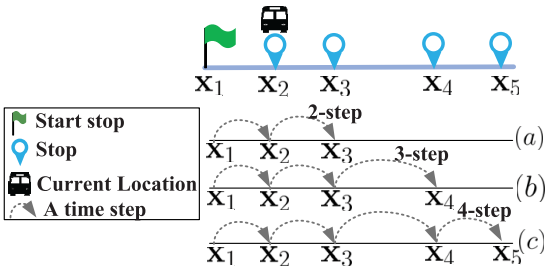


Fig. 5. The illustration of unrolling RNN in (12). (a), (b), and (c) predict the arrival times for the stops $\mathbf{x}_3, \mathbf{x}_4$, and $\mathbf{x}_5$, respectively.

in which

$$\text{smooth}_{\ell_1}(x) = \begin{cases} \frac{1}{2}x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwie,} \end{cases} \tag{11}$$

is a robust ℓ_1 loss that is less sensitive to outliers than the ℓ_2 one. When the regression targets are unbounded, training with ℓ_2 loss requires to tune learning rates carefully, so as to prevent exploding gradients. (11) eliminates this sensitivity.

C. Prediction by Unrolling RNN

Once a LSTM RNN network for a route is trained, prediction amounts to unrolling RNN along time steps. Concretely, given the current location of a bus p_k, and its heterogeneous measurement $\mathbf{x}_k$, we recursively predict the arrival times for the multiple ahead stops $t_{k+\Delta}$, $\Delta = 1, \ldots, K-k$ as follows:

$$\hat{t}_{k+1}, c_k, h_k \xleftarrow{predict} RNN\left(c_{k-1}, h_{k-1}, \mathbf{x}_k\right), \tag{12}$$

$$\mathbf{x}_{k+1} \xleftarrow{feature} Coding\left(\hat{t}_{k+1}, d_{k+1}, d_{k+2}, r_{k+1}\right), \tag{13}$$

where $RNN(\cdot)$ is the prediction model from (3) to (9), and $Coding(\cdot)$ is the feature coding in (2). According to the unrolling process of RNN, these stops $p_{k+1}, \ldots, p_{k+\Delta-1}$ serve as the "jumping" points for the stop $p_{k+\Delta}$. Fig. 5 graphically shows the prediction process of (12). It can be observed that the number of the unrolled steps for the latter stops is gradually increasing. That is, for the different stops, the number of the time step is totally different in our framework.

In order to obtain the fine-grained traffic conditions, we partition the travel distance into segments; besides, in order to obtain a better result, the locations of intersections for a route are incorporated into (2). However, densely sampled jumping points do not necessarily bring any gains of accuracy, since the number of the dependencies for each prediction is determined by the physic property of each stop. As a result, the stops, the intersections, and the interpolated points constitute the "jumping" points for an ahead stop. Essentially, the "jumping" points serve as sensors to encode the physic property of local road segments for prediction. In this paper, we empirically divide a trace into road segments every 200 meters.

In order to organize these heterogeneous measurements into a vector, bus stop d, intersection s, and interpolated point l should be distinguished. One-hot coding [40] therefore is introduced. That is, $[0, 0, 1]^T$ is for stop d, $[0, 1, 0]^T$ is

TABLE II

ASSIGNING $\tilde{r}_k, \tilde{d}_k, \tilde{d}_{k+1}$ AND THE CORRESPONDING ONE-HOT CODING IN (14)

The k-th jumping point $\rightarrow$ The $k+1$-th one	$\tilde{r}_k$	$\tilde{d}_k$	$\tilde{d}_{k+1}$	One-hot coding
Stop $\rightarrow$ Intersection / interpolated point	r_k	d_k	s_{k+1}/l_{k+1}	$[0,0,1]$
Stop $\rightarrow$ Stop	r_k	d_k	d_{k+1}	$[0,0,1]$
Intersection / interpolated point $\rightarrow$ Intersection / interpolated point	0	s_k/l_k	s_{k+1}/l_{k+1}	$[0,1,0]/[1,0,0]$
Intersection / interpolated point $\rightarrow$ Stop	0	s_k/l_k	d_{k+1}	$[0,1,0]/[1,0,0]$

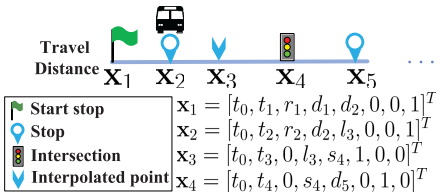


Fig. 6. A example of (14).

for intersection s, and $[1,0,0]^T$ is for interpolated point l. Consequently, $Coding(\cdot)$ function for the k-th "jumping" point in (13) is updated as follows:

$$\mathbf{x}_k = \left[t_0, t_k, \tilde{r}_k, \tilde{d}_k, \tilde{d}_{k+1}, \text{one hot coding}\right]^T, \tag{14}$$

where $\tilde{d}_k$ and $\tilde{d}_{k+1}$ represent two consecutive "jumping" points, and $\tilde{r}_k$ measures the scale of "jumping" point. Table II lists how to assign these variables for one-hot coding. A toy example is illustrated in Fig. 6.

Note that (14) is a general scheme to encode the heterogeneous measurements. Principally, if the other factors (such as, weather, and traffic conditions) are encoded into a feature vector, one-hot coding scheme in (14) would efficiently fuse these factors into a learning system.

D. Implementations

1) Initiations and Mini-Batch Training: Before training, the initializations of both W_* and b_* are crucial to guarantee a good performance. The matrices W_* are all initialized from the zero-mean Gaussian distributions with the standard deviations $1/(N_{in}+N_{out})$, where N_{out} and N_{in} are the size of the current layer and the previous layer, respectively [41]. Biases $b_{\{i,j,o\}}$ are initialized as 0, and b_f is initialized as 5. The network setting is: $W_{ix}, W_{jx}, W_{fx}, W_{ox} \in \mathbb{R}^{8\times 64}$, $W_{ih}, W_{jh}, W_{fh}, W_{oh} \in \mathbb{R}^{64\times 64}$, $W_{th} \in \mathbb{R}^{64\times 1}$. These parameters are trained by Stochastic Gradient Descent (SGD) with a momentum of 0.95 and a learning rate of 0.0001 by Back Propagation Through Time (BPTT) [19] on a Nvidia GeForce GTX 980.

During the training, each SGD mini-batch is constructed by 256 sequences which are uniformly picked at random (as a common practice, we actually iterate over the permutations of a data set). We run SGD for 40k mini-batch iterations with the learning rate 0.0001, and then lower the learning rate to 0.00001 for another 10k iterations.

2) Scale Problem: Entries in the vector $\mathbf{x}_k$ have different scales. We have explored two ways for this problem: (1) via the zero-mean normalization,

$$\mathbf{x}_{k_i} \leftarrow \frac{\mathbf{x}_{k_i} - \mu_i}{\sigma_i}, \tag{15}$$

where $\mathbf{x}_{k_i}$ is the ith entry of the vector $\mathbf{x}_k$, μ_i and σ_i are the mean and the standard deviation of $\{\mathbf{x}_{1_i}, \ldots, \mathbf{x}_{K-1_i}\}$, respectively; and (2) via the max-mean normalization as follows,

$$\mathbf{x}_{k_i} \leftarrow \mathbf{x}_{k_i} - \max_k \left(\mathbf{x}_{k_i}\right), \tag{16}$$

$$\mathbf{x}_{k_i} \leftarrow \frac{\mathbf{x}_{k_i}}{\mu_i}. \tag{17}$$

We have empirically found that both methods achieve the similar performances. In the following results, the zero-mean normalization is used for its simplicity.

V. EXPERIMENT AND DISCUSSION

A. Evaluations and Experimental Setup

To evaluate the overall performance of each method for a route, Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Mean Absolute Percentage Error (MAPE) are utilized as follows:

$$MAE = \frac{2\sum_{i=1}^{N}\sum_{k=1}^{K-1}\sum_{\Delta=1}^{K-k}\left|t^i_{k+\Delta} - \hat{t}^i_{k+\Delta}\right|}{NK(K-1)},$$

$$RMSE = \frac{\sum_{i=1}^{N}\sum_{k=1}^{K-1}\sqrt{\frac{1}{K-k}\sum_{\Delta=1}^{K-k}\left(t^i_{k+\Delta} - \hat{t}^i_{k+\Delta}\right)^2}}{N(K-1)},$$

$$MAPE = \frac{2\sum_{i=1}^{N}\sum_{k=1}^{K-1}\sum_{\Delta=1}^{K-k}\left|\frac{t^i_{k+\Delta} - \hat{t}^i_{k+\Delta}}{t^i_{k+\Delta}}\right|}{NK(K-1)} \times 100\%,$$

where the superscript i means the i-th trip for a route, and N is the number of trips for this route. MAE only measures the absolute errors between the prediction and the target; MAPE measures the relative errors by normalization; RMSE measures the average absolute squared errors between the prediction and the target.

We also analyze the distributions of the absolute errors, $|t^i_{k+\Delta} - \hat{t}^i_{k+\Delta}|$, along the travel distance [34]. Concretely, the distance space is partitioned into two kilometer bins $[0,2), [2,4), [4,6), \ldots$, *etc*. The distributions of errors are compared by boxplot which is a convenient way to vividly describe the distributions of data through their quartiles. The boxplot in this paper displays the median (*i.e.*, middle line inside the box), the first and third quartiles (*i.e.*, the bottom and the top of the box).

TABLE III
COMPARISONS AMONG STATE-OF-THE-ART METHODS ON OUR DATASET

Method	<10KM (19 Routes)			10-15KM (17 Routes)			>15KM (11 Routes)		
	MAE(min.)	RMSE(min.)	MAPE(%)	MAE(min.)	RMSE(min.)	MAPE(%)	MAE (min.)	RMSE (min.)	MAPE(%)
k-NN	1.31±0.22	1.52±0.27	18.16±1.98	1.49±0.24	1.74±0.28	17.26±1.73	1.43±0.11	1.66±0.14	16.00±1.35
KR	1.23±0.20	1.42±0.25	17.29±1.82	1.41±0.22	1.64±0.26	16.59±1.75	1.38±0.15	1.61±0.19	15.71±1.36
SPB	1.50±1.51	1.80±0.64	48.10±41.58	2.51±0.78	3.06±0.93	44.79±15.21	2.78±0.84	3.51±1.05	60.20±49.84
LRT	1.21±0.19	1.40±0.23	**17.28±2.35**	1.37±0.22	1.60±0.26	16.21±1.97	1.35±0.12	1.57±0.14	15.64±1.65
AMM	1.29±0.30	1.47±0.34	31.17±15.81	1.55±0.34	1.78±0.38	26.40±21.65	1.44±0.25	1.68±0.29	36.31±55.43
KFP	2.19±0.94	2.31±0.98	84.69±59.19	3.07±1.16	3.23±1.18	78.63±38.53	2.68±0.40	2.84±0.39	71.27±39.16
MLP	1.22±0.34	1.37±0.39	28.59±12.35	1.70±0.33	1.96±0.38	23.01±3.71	1.95±0.64	2.30±0.95	23.07±5.78
LSTM RNN	**0.93±0.36**	**1.06±0.42**	18.66±4.05	**1.23±0.25**	**1.41±0.28**	**15.92±1.85**	**1.19±0.13**	**1.36±0.17**	**14.38±2.67**
Improvement	23.14%	22.63%	-7.99%	10.22%	11.88%	1.79%	11.85%	13.38 %	8.05%

B. Methods in Comparison Study

Our specific experimental goal is to compare the proposed approach with the state-of-the-art methods:

1. **k-NN [33] or Kernel Regression (KR) [32].** This baseline belongs to the searching-based method. To achieve the non-linear ability, the weights in [32] were computed as Gaussian kernel, *i.e.*, $\exp(-\|x-y\|^2/b)$, where x and y represent traces, and b is the bandwidth of the kernel. During the experiments, following [32] and [33], we set $b=1$ for KR and $k=10$ for k-NN, respectively.
2. **SVM on Probe Buses (SPB) [8].** This baseline belongs to the regression-based method. Reference [8] firstly divided a trip into road segments, and then proposed four features based on the probe buses. Comparing with this baseline demonstrates that our approach can achieve superior results without any probe information. SVM is implemented by the open source package.[3]
3. **Linear Regression on Trajectories (LRT) [32].** This baseline belongs to the regression-based method. By assuming that the travel time follows a multivariate Gaussian distribution, the predicted arrival time is the posterior mean value. For further details, please refer to [32].
4. **Additive Mixed Model (AMM) [34].** This baseline belongs to the regression-based method. Reference [34] used the additive model to correlate multiple factors, such as, the weekend, the current time, the travel distances. Comparing with this baseline demonstrates that our approach can efficiently exploit the multiple measurements. Following [34], AMM is implemented as the cubic representation splines, and the tensor product smooths with 5 knots.[4]
5. **Kalman Filter with Probe buses (KFP) [15].** This baseline belongs to the filtering based method. Reference [15] firstly divided a travel distance into road segments, and then learned the dynamics of KF. However, in our data set, the time headway of buses on a route ranges from 10 to 20 minutes. Therefore, the dynamics learned from the probe buses achieved inferior performances in our unreported results. In our implementation, the linear dynamics were estimated with expectation maximization algorithm [42].
6. **Multilayer Perceptron (MLP) [43].** This baseline belongs to the regression-based method. Reference [43] used four features, *i.e.*, the time of day interval, the locations of bus stops, the arrival time, for a MLP with three layers. By the brute-force approach, we set the number of nodes in the hidden layer to be 15. MLP was implemented with the function `newff` in MATLAB.

Neither the source codes nor the test data sets are supplied by these papers; besides, several approaches (*i.e.*, SPB [8], AMM [34], KFP [15], and MLP [43]) involve many engineering details that have not been clearly presented in the papers. Therefore, we tried our best to re-implement their approaches on our data set. Due to the fact that the length of this paper is limited, more details about these methods (*e.g.*, the hyper-parameters, the training methods, and the implementation tricks) are summarized in the supplementary material.

C. Comparisons With the State-of-The-Arts

Table. III shows the comparisons among different methods. Both the mean and the standard deviation of each approach in Section V-B are computed for each route.[5] In Table. III, LSTM RNN achieves the smallest MAE, RMSE, MAPE for all groups but the "<10 KM" one for MAPE. The performances of LSTM RNN slightly vary with respect to the length of a route. For instance, LSTM RNN achieves 1.23 and 1.19 MAEs in the "10−15 KM" and the ">15 KM" groups, respectively.

KFP [15] achieves the worst performance than the other methods. The main explanation is that KFP [15], a filter-based approach, only depends on the "correctness" from the consecutive time step. Moreover, if the linear dynamics learned from the history data are different from the current traffic conditions, the prediction system would rapidly deteriorate. Both k-NN and KR achieves very similar performances which are inferior to our approach, either. As expected, the search-based approach tends to achieve inferior results when traffic conditions cannot re-occur for the current prediction. Interestingly, the regression-based approach, (*i.e.*, SPB [8], LRT [32], and AMM [34]), achieves a similar result. The explanation is that LRT, AMM and SPB avoid the difficulty of MSAP by iteratively bootstrapping along time steps. In contrast, our approach starts with the heterogeneous measurements and then builds the long-range dependencies among time steps.

[3]http://www.csie.ntu.edu.tw/~cjlin/libsvm/

[4]https://cran.r-project.org/web/packages/mgcv/

[5]The detailed results for each route is presented in the supply material.

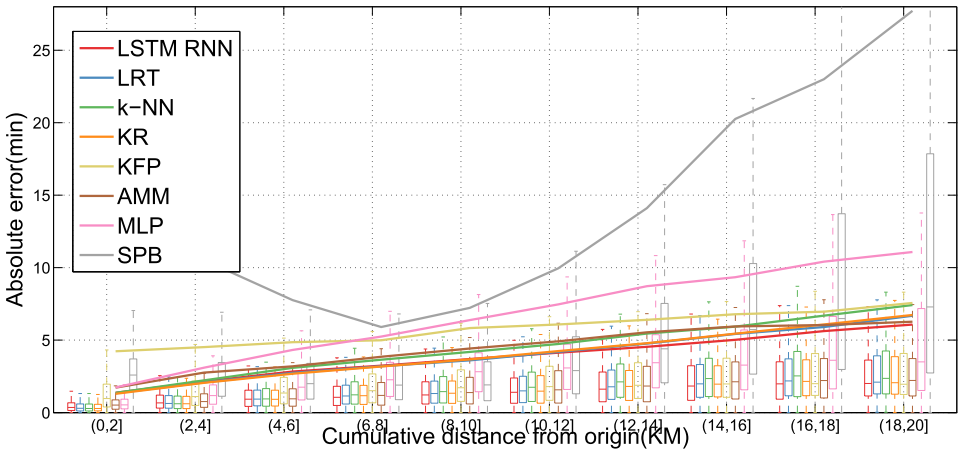


Fig. 7. Comparisons among boxplots of absolute prediction errors of 6 methods and of our approach. Note that the line represent 95th percentiles of absolute errors (best viewed in color).

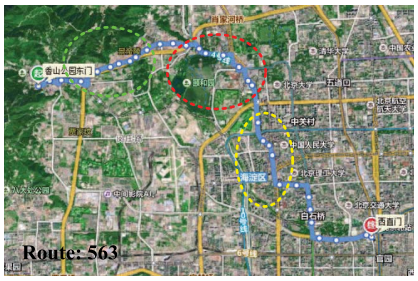


Fig. 8. The spatial distribution of route 563 (best viewed in color).

Therefore, the performance of our approach is more robust and superior than the other ones.

To better illustrate the power of LSTM RNN, the relative improvement over the best results is computed as: $\frac{LSTMRNN-Best}{Best} \times 100\%$, where $LSTMRNN$ and $Best$ are the results achieved by LSTM RNN and the state-of-the-arts, respectively. Table III shows that LSTM RNN improves at least 10% with respect to both MAE and RMSE for all three groups.

To give a more vivid view of the results, Fig. 7 shows the boxplot of the absolute errors for route 563 which will be detailed in Section V-D. Fig. 7 illustrates that LSTM RNN outperforms the other state-of-the-art methods in all bins except for the [0,2) bin and the [2,4) one. The explanation is that both c_k and h_k in LSTM RNN cannot fully unleash the power of the long-rang dependencies when a few time steps are observed. Principally, the more number of time steps LSTM RNN observes, the more accurate both the states c_k and h_k are. As a result, although the performance of LSTM RNN is slightly worse than KR [32] (with the median value < 0.5 min.) at the both bins [0,2) and [2,4), LSTM RNN statistically outperforms all the other methods in the other longer distance bins.

D. Revisit Our Method

In this section, the route 563 with the longest travel distance is selected as a case study to analyze our method. Fig. 8 illustrates that the route 563 crosses the centralized business area (*i.e.*, Zhongguan Cun) indicated by the yellow circle, the famous historical sight (*i.e.*, Summer Palace) indicated by the red one, and the tourist attraction (*i.e.*, Fragrance Hill) indicated by the green one. Therefore, route 563 is very challenging for the bus arrival time prediction task.

TABLE IV
EFFECTS OF VARIOUS CHOICES ON LSTM RNN PERFORMANCE

"Jumping" points	Choices		
Stops?	✓	✓	✓
Interpolated points?		✓	✓
Intersections?			✓
MAE (min)	1.38	1.21	**1.18**
RMSE (min)	1.61	1.44	**1.40**
MAPE(%)	17.01	11.93	**11.75**

1) The Gains of Heterogeneous Measurements: To understand LSTM RNN better, we have examined how each measurement affects the final performance. For all of the following experiments, we exactly used the same setting in Section IV-D, except for the variable measurement.

a) Interpolated points are crucial: Table IV shows that we can improve 0.17(= 1.38−1.21) MAE, 0.17(= 1.61−1.44) RMSE, and 5.08%(= 17.01−11.93) MAPE via this sampling strategy. Because these points reflect the physical properties of the road segments. However, without considering these physical properties, simply up-sampling the road segments does not necessarily bring more gains of accuracy. Table V further compares the performances between the different up-sampling rates. "50M", "100M", and "200M" in Table V means that a travel distance is divided into road segments every 50, 100, and 200 meters, respectively. Thus, the up-sampling rate of "50M" is the 4 times as dense as "200M". Unexpectedly, "50M" only achieves a slightly better results than that of "200M". It verifies that RNN needs more physically meaningful road segments, *e.g.*, intersections, rather than attempting to increase the accuracies by densely up-sampling.

b) More heterogeneous measurements are better: From Table IV, we can see that LSTM RNN can achieve better results, if more measurements are used. For instance, MAE

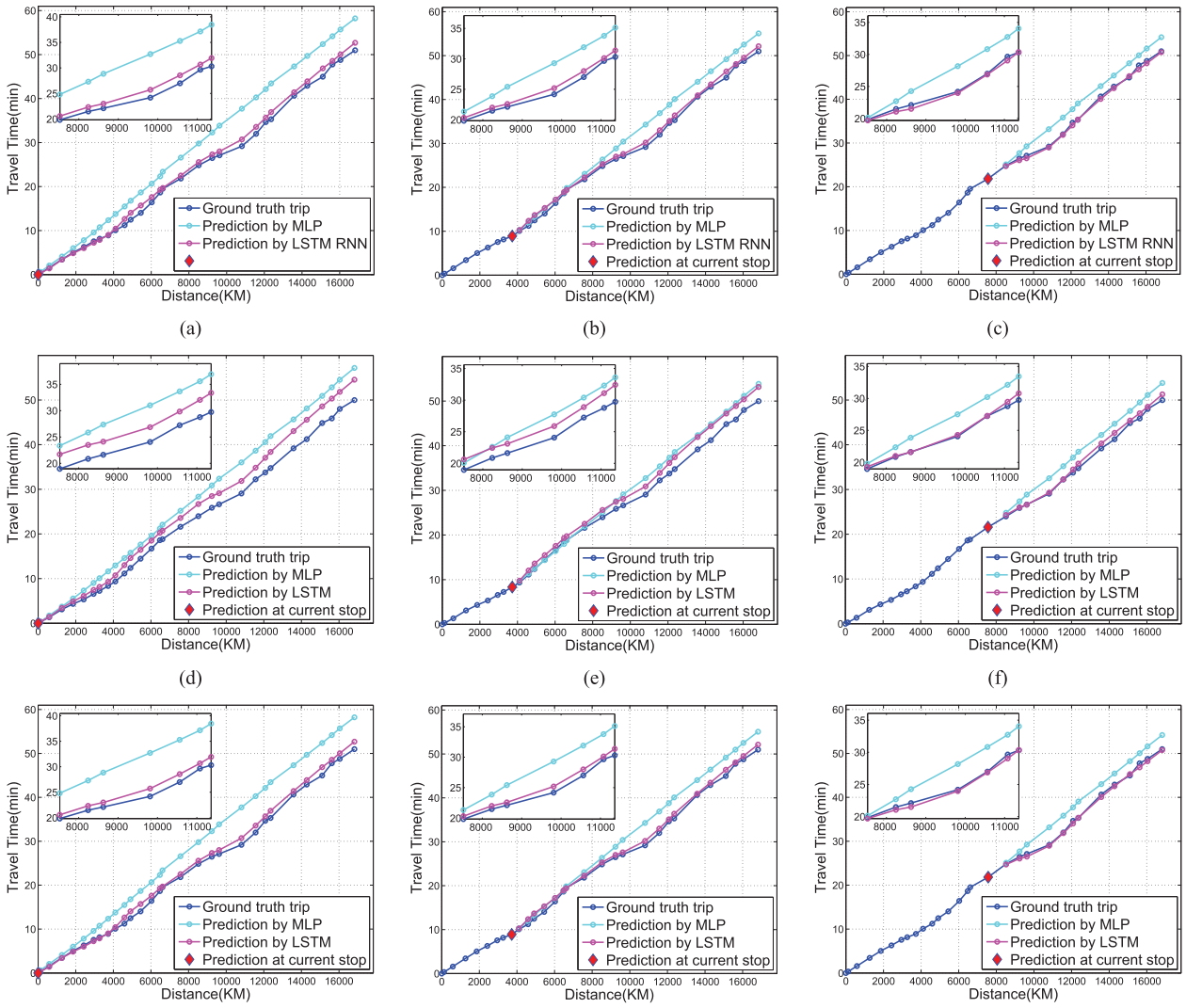


Fig. 9. The comparisons between with and without the long-range dependencies (best viewed in color). (7:41) in the first row (a)-(c) means that a bus departed from the start stop at 7:41; (13:27) in the second rows (d)-(f) illustrates that a bus departed at 13:27; (17:41) int the last row (g)-(i) shows that a bus departed at 17:41.

TABLE V
EFFECTS OF UPSAMPLING RATES ON LSTM RNN PERFORMANCES

Upsampling rate	MAE(min.)	RMSE (min.)	MAPE(%)	Time cost per. Stop† (ms.)
50M	1.19	1.42	12.08	6.8
100M	**1.18**	**1.40**	**11.75**	1.3
200M	1.20	1.43	12.11	**0.5**

† Here, LSTM RNN was implemented by Matlab on a CPU with 3.3 GHz.

is improved from 1.38 to 1.18 when the locations of stops, interpolated points, and intersections are all used in LSTM RNN. This result also matches our intuition that the complex correlations among the multiple measurements bring a better prediction result.

2) Why Long-Range Dependencies Are Needed?: RNN would be unrolled into a forward neural network, if there is no any feedback involved. Therefore, for the comparison between the short dependencies (*i.e.*, two time steps) and the long dependencies (*i.e.*, multiple time steps), we use MLP as a typical forward neural network to build the short dependencies. During the training of MLP, the feature in (14) is used to guarantee a fair comparison with LSTM RNN.

Firstly, three ground truth trips were randomly selected from the following time slices: (1) 7:30-9:30 when the roads have the severe traffic congestion, (2) 11:30-13:30 when traffic becomes fluent, and (3) 17:30-19:30 when the roads have the severe traffic congestion. Next, three bus stops are selected for comparisons: (1) the start stop (*i.e.*, East Gate of the Fragrance Hill Park), (2) the 8-th stop (*i.e.*, Fenghu Barrack) and (3) the 16-th stop (*i.e.*, North Gate of Summer Palace). The 8-th stop and the 16-th one are the leaving stop of the tourist attraction sight and the approaching stop for the famous historical sight, respectively. These stops are thus selected to examine the robustness of these prediction models.

From Figs. 9(a) to (i), we observe that LSTM RNN outperforms its counterpart, *i.e.*, MLP. These results indicate a consistent positive effect of the long-range dependencies from the multiple time steps. Moreover, LSTM RNN achieves

very robust performances across different time slices, *i.e.*, the curves from the second column ((b), (e), and (h)), and the curves from the third column ((c), (f), and (i)). In addition, there are two interesting observations in Fig. 9:

- Although LSTM RNN achieves the unsatisfied results at the start stop, the long-range dependencies would rectify the results if a few time steps have been observed by LSTM RNN. For instance, the curves of LSTM RNN at the 16-th stop in Figs. 9(c), 9(f), and 9(i) are more closer to the ground truth than that of LSTM RNN at the start stop in Figs. 9(a), 9(d), and 9(g). It indicates that the long-range dependencies propagate the observations from the passed stops to the ahead ones for prediction.
- Unexpectedly, the predictions by MLP are nearly linear with respect to the trip distances. Considering the differences between MLP and LSTM RNN, the only explanation is that the long-range dependencies are crucial to achieve the non-linear prediction ability for the MSAP task.

VI. Conclusions

In this paper, for the bus arrival time prediction, we have described a learning-based approach to leverage the long-range dependencies among the multiple time steps. The proposed LSTM RNN treats bus arrival time prediction as the MSAP task. The thorough analysis has revealed that the MSAP task should incorporate the long-range dependencies for prediction. This is an important finding that has not been reported in the literatures previously. Moreover, this finding is examined on the largest data set which is freely available for all researchers. This is one of the main contributions of this paper, too.

Although the weak spatio-temporal features in (2) are used, RNN with the LSTM block achieves the best performance, relatively surpassing the state-of-the-art methods at least 10% MAE and 10% RMSE. These results indicate that the long-range dependencies are crucial to achieve the non-linear ability for prediction. The proposed prediction method is one of the contributions made here.

Introducing one-hot coding to encode the heterogeneous measurements is another contribution of this paper. Simply by adopting one-hot coding, multiple features are easily encoded into a vector space. This merit gracefully combines more heterogeneous measurements into a learning system.

In the future, we aim to develop this approach by incorporating more heterogeneous data, such as the automatic passenger counter, automated fare collection, weather, and traffic conditions. Moreover, considering bus routes essentially affect each other, how to incorporate the spatial relationship between bus stops into our prediction model is an interesting direction.

References

[1] UNP Division. (2015). *Wold Population Prospects*. [Online]. Available: https://esa.un.org/unpd/wpp/Publications/Files/WPP2015_Methodology.pdf

[2] I. Psarros, K. Kepaptsoglou, and M. G. Karlaftis, "An empirical investigation of passenger wait time perceptions using hazard-based duration models," *J. Public Transp.*, vol. 14, no. 3, pp. 109–122, 2011.

[3] L. Bailey, *Public Transportation and Petroleum Savings in the U.S.: Reducing Dependence on Oil*. Fairfax, VA, USA: ICF International, 2007.

[4] M. Salek and R. Machemehl, "Characterizing bus transit passenger wait times," Dept. Center Transp. Res., Univ. Texas, Austin, TX, USA, Tech. Rep. SWUTC/99/167211-1, Jun. 1999.

[5] K. Watkins, B. Ferries, A. Borning, G. Rutherford, and D. Layton, "Where is my bus? Impact of mobile real-time information on the perceived and actual wait time of transit riders," *Transp. Res. A, Police Pract.*, vol. 45, no. 8, pp. 839–848, 2011.

[6] Q. Chen, E. Adida, and J. Lin, "Implementation of an iterative headway-based bus holding strategy with real-time information," *Public Transp.*, vol. 3, no. 4, pp. 165–186, 2013.

[7] E. Holroyd and D. Scraggs, "Waiting times for buses in central London," *Traffic Eng. Control*, vol. 3, no. 8, pp. 158–160, 1966.

[8] B. Yu, W. H. K. Lam, and M. L. Tam, "Bus arrival time prediction at bus stop with multiple routes," *Transp. Res. C, Emerg. Technol.*, vol. 19, no. 6, pp. 1157–1170, Dec. 2011.

[9] M. Altinkaya and M. Zontul, "Urban us arrival time predication: A review of computational models," *Int. J. Recent Technol. Eng.*, vol. 2, no. 4, pp. 164–169, 2013.

[10] S. Zadeh, T. Awar, and M. Basirat, "A survey on application of artificial intelligence for bus arrival time prediction," *J. Theory Appl. Inf.*, vol. 1, no. 46, pp. 516–525, 2012.

[11] D. Sun, H. Luo, L. Fu, W. Liu, X. Liao, and M. Zhao, "Predicting bus arrival time on the basis of global positioning system data," *Transp. Res. Rec., J. Transp. Res. Board*, no. 2034, pp. 62–72, 2007.

[12] A. O'Sullivan, F. Pereira, J. Zhao, and H. Koutsopoulos, "Unvertainty in bus arrival time predictions: Treating heteroscedasticity with a meta-modal approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 11, pp. 3286–3296, Nov. 2016.

[13] S. I.-J. Chien and C. M. Kuchipudi, "Dynamic travel time prediction with real-time and historic data," *J. Transp. Eng.*, vol. 129, no. 6, pp. 608–616, 2003.

[14] M. D'Angelo, H. Al-Deek, and M. Wang, "Travel-time prediction for freeway corridors," *Transp. Res. Rec., J. Transp. Res. Board*, no. 1676, pp. 184–191, Jan. 1999.

[15] L. Vanajakshi, S. Subramanian, and R. Sivanandan, "Travel time prediction under heterogeneous traffic conditions using global positioning system data from buses," *IET Intell. Transp. Syst.*, vol. 3, no. 1, pp. 1–9, 2009.

[16] Y. Bin, Y. Zhongzhen, and Y. Baozhen, "Bus arrival time prediction using support vector machines," *J. Intell. Transp. Syst.*, vol. 10, no. 4, pp. 151–158, 2006.

[17] S. I.-J. Chien, Y. Ding, and C. Wei, "Dynamic bus arrival time prediction with artificial neural networks," *J. Transp. Eng.*, vol. 128, no. 5, pp. 429–438, 2002.

[18] B. Williams and L. Hoel, "Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: Theoretical basis and empirical results," *J. Transp. Eng.*, vol. 129, no. 6, pp. 664–672, 2003.

[19] R. Williams and J. Peng, "An efficient gradient-based algorithm for on-line training of recurrent network trajectories," *Neural Comput.*, vol. 2, no. 4, pp. 491–501, 1990.

[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[21] J. Chung, C. Gulcehre, K. H. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *Proc. Conf. Neural Inf. Process. Syst., Workshop Deep Learn.*, 2014, pp. 1–8.

[22] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.

[23] C.-H. Wu, J.-M. Ho, and D. T. Lee, "Travel-time prediction with support vector regression," *IEEE Trans. Intell. Transp. Syst.*, vol. 5, no. 4, pp. 276–281, Dec. 2004.

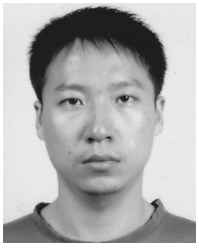
[24] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines*. Cambridge, U.K.: Cambridge Univ. Press, 2000.

[25] Y. Ramakrishna, P. Ramakrishna, V. Lakshmanan, and R. Sivanandan, "Use of GPS probe data and passenger data for prediction of bus transit travel time," in *Proc. Transp. Land Use, Planning, Air Qual. Congr.*, 2008, pp. 124–133.

[26] M. Hagan, H. Demuth, and M. Beale, *Neural Network Design*. Boston, MA, USA: PWS, 1996.

[27] J. Van Lint, S. P. Hoogendoorn, and H. J. van Zuylen, "Accurate freeway travel time prediction with state-space neural networks under missing data," *Transp. Res. C, Emerg. Technol.*, vol. 13, nos. 5–6, pp. 347–369, Oct./Dec. 2005.

[28] H. Liu, K. Zhang, R. He, and J. Li, "A neural network model for travel time prediction," in *Proc. IEEE Int. Conf. Intell. Comput. Intell. Syst.*, Nov. 2009, pp. 752–756.
[29] G. Abu-Lebdeh and A. Singh, "Modeling arterial travel time with limited traffic variables using conditional independence & State-space neural networks," in *Proc. Int. Symp. Highway Capacity Qual. Service*, 2011, pp. 207–217.
[30] J. M. Zamarreño and P. Vegab, "State space neural network. properties and application," *Neural Netw.*, vol. 11, no. 6, pp. 1099–1112, 1998.
[31] T. Liu, J. Ma, W. Guan, Y. Song, and H. Niu, "Bus arrival time prediction based on the *k*-nearest neighbor method," in *Proc. Int. Joint Conf. Comput. Sci. Optim.*, 2012, pp. 480–483.
[32] M. Sinn, J. Yoon, F. CalaBrese, and E. Bouillet, "Predicting arrival times of buses using real-time gps measurements," in *Proc. Int. IEEE Conf. Intell. Transp. Syst.*, Sep. 2012, pp. 1227–1232.
[33] C. Coffey, A. Pozdnoukhov, and F. CalaBrese, "Time of arrival predictability horizons for public bus routes," in *Proc. ACM SIGSPATIAL Int. Workshop Comput. Transp. Sci.*, 2011, pp. 1–5.
[34] M. Kormaksson, L. Barbosa, M. Vieira, and B. Zadrozny, "Bus travel time predictions using additive models," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2014, pp. 875–880.
[35] X. Ma and Y. Wang, "Development of a data-driven platform for transit performance measures using smart card and GPS data," *J. Transp. Eng.*, vol. 140, no. 12, pp. 401–416, 2014.
[36] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, 2006.
[37] H. Miura, "A study of travel time prediction using universal Kriging," *Top*, vol. 18, no. 1, pp. 257–270, 2010.
[38] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Trans. ASME, D, J. Basic Eng.*, vol. 82, pp. 35–45, 1960.
[39] D. Seborg, T. Edgar, D. Mellichamp, and F. Doyle, *Process Dynamics and Control*. Hoboken, NJ, USA: Wiley, 2011.
[40] D. Harris and S. Harris, *Digital Design and Computer Architecture*, 2nd ed. San Francisco, CA, USA: Morgan Kaufmann, 2012.
[41] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *J. Mach. Learn. Res.*, vol. 9, pp. 249–256, May 2010.
[42] V. Digalakis, J. R. Rohlicek, and M. Ostendorf, "ML estimation of a stochastic linear system with the EM algorithm and its application to speech recognition," *IEEE Trans. Speech Audio Process.*, vol. 1, no. 4, pp. 431–442, Oct. 1993.
[43] Z. Gurmu and W. Fan, "Artifical neural network travel time prediction model for buses using only GPS data," *J. Public Transp.*, vol. 7, no. 2, pp. 45–65, 2014.



Junbiao Pang received the B.S. and M.S. degrees in computational fluid dynamics and computer science from the Harbin Institute of Technology, Harbin, China, in 2002 and 2004, respectively, and the Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2011.

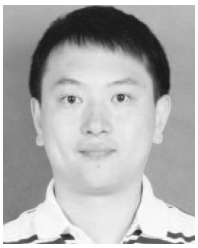
He is currently an Associate Professor with the Faculty of Information, Beijing University of Technology, Beijing. He has authored or co-authored approximately 30 academic papers in publications, such as the IEEE TRANSACTIONS ON IMAGE PROCESSING and ACM *Multimedia*. His research interests include multimedia and machine learning.

Jing Huang received the B.S. degree in computer science and technology from the North China University of Technology, Beijing, China, in 2014, and the M.S. degree in computer science and technology from the Beijing University of Technology, Beijing, in 2017.

She is currently a Software Developer with IBM China Investment Co. Ltd. Her research interests include machine learning in cloud platform and the applications of artificial intelligence.

Yong Du received the B.S. degree in radio technology from Beijing Union University, Beijing, China, in 1995, and the M.S. degree in computer application technology from the Beijing University of Technology, Beijing, in 2003.

He is currently the Secretary and the Associate Director of the Beijing Transportation Information Center, Beijing, and the legal person of the Beijing Intelligent Transportation Association. His research interests include the application of key technologies for intelligent transportation.



Haitao Yu received the B.S. degree in management information systems from the Beijing Information Science and Technology University, Beijing, in 2005, and the M.S. degree in computer software from Beihang University, Beijing, in 2009, where he is currently pursuing the Ph.D. degree in computer science and technology.

He is currently the Associate Dean of the Beijing Transportation Information Center, Beijing. His research interests include traffic data analysis for key technologies in traffic information service.

Qingming Huang (SM'08–F'18) received the bachelor's degree in computer science and the Ph.D. degree in computer engineering from the Harbin Institute of Technology, China, in 1988 and 1994, respectively. He is currently a Professor with the University of Chinese Academy of Sciences and an Adjunct Research Professor with the Institute of Computing Technology, Chinese Academy of Sciences.

He has authored or co-authored over 400 academic papers in prestigious international journals and top-level international conferences. His research areas include multimedia computing, image processing, computer vision, and pattern recognition. He has served as the General Chair, Program Chair, Track Chair, and a TPC Member for various conferences, including ACM Multimedia, CVPR, ICCV, ICME, and PSIVT. He is an Associate Editor for the IEEE TRANSACTIONS ON CSVT and *Acta Automatica Sinica* and the Reviewer of various international journals, including the IEEE TRANSACTIONS ON MULTIMEDIA and the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY.

Baocai Yin received the M.S. and Ph.D. degrees in computational mathematics from the Dalian University of Technology, Dalian, China, in 1988 and 1993, respectively.

He is currently a Professor with the Department of Electronic Information and Electrical Engineering, Dalian University of Technology. He is also the Director of the Beijing Key Laboratory of Multimedia and Intelligent Software Technology, Beijing, China. He has authored or co-authored over 200 academic papers in prestigious international journals, including the IEEE TRANSACTIONS ON MULTIMEDIA and the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, and top-level conferences, such as INFOCOM and ACM SIGGRAPH. His research areas include multimedia, image processing, computer vision, and pattern recognition.

Dr. Yin is currently the Editorial Member for the *Journal of Information and Computational Science* (USA).