MetaTP: Traffic Prediction with Unevenly-Distributed Road Sensing Data via Fast Adaptation

WEIDA ZHONG, University at Buffalo, USA
QIULING SUO, University at Buffalo, USA
ABHISHEK GUPTA, University at Buffalo, USA
XIAOWEI JIA, University of Pittsburgh, USA
CHUNMING QIAO, University at Buffalo, USA
LU SU*, Purdue University, USA

With the popularity of smartphones, large-scale road sensing data is being collected to perform traffic prediction, which is an important task in modern society. Due to the nature of the roving sensors on smartphones, the collected traffic data which is in the form of multivariate time series, is often temporally sparse and unevenly distributed across regions. Moreover, different regions can have different traffic patterns, which makes it challenging to adapt models learned from regions with sufficient training data to target regions. Given that many regions may have very sparse data, it is also impossible to build individual models for each region separately. In this paper, we propose a meta-learning based framework named MetaTP to overcome these challenges. MetaTP has two key parts, i.e., basic traffic prediction network (base model) and meta-knowledge transfer. In base model, a two-layer interpolation network is employed to map original time series onto uniformly-spaced reference time points, so that temporal prediction can be effectively performed in the reference space. The meta-learning framework is employed to transfer knowledge from source regions with a large amount of data to target regions with a few data examples via fast adaptation, in order to improve model generalizability on target regions. Moreover, we use two memory networks to capture the global patterns of spatial and temporal information across regions. We evaluate the proposed framework on two real-world datasets, and experimental results show the effectiveness of the proposed framework.

CCS Concepts: • **Human-centered computing** → **Ubiquitous and mobile computing systems and tools**; • **Networks** → *Sensor networks.*

Additional Key Words and Phrases: traffic prediction, meta learning

ACM Reference Format:
Weida Zhong, Qiuling Suo, Abhishek Gupta, Xiaowei Jia, Chunming Qiao, and Lu Su. 2021. MetaTP: Traffic Prediction with Unevenly-Distributed Road Sensing Data via Fast Adaptation. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 5, 3, Article 141 (September 2021), 28 pages. https://doi.org/10.1145/3478083

[*] Lu Su is the corresponding author.

Authors' addresses: Weida Zhong, University at Buffalo, New York, USA, weidazho@buffalo.edu; Qiuling Suo, University at Buffalo, New York, USA, qiulings@buffalo.edu; Abhishek Gupta, University at Buffalo, New York, USA, agupta22@buffalo.edu; Xiaowei Jia, University of Pittsburgh, Pennsylvania, USA, xiaowei@pitt.edu; Chunming Qiao, University at Buffalo, New York, USA, qiao@buffalo.edu; Lu Su, Purdue University, Indiana, USA, lusu@purdue.edu.

2474-9567/2021/9-ART141 $15.00
https://doi.org/10.1145/3478083

1 INTRODUCTION

Traffic prediction plays an important role in modern society. Effective traffic prediction can directly alleviate the issue of traffic congestion, which costs each American almost 100 hours a year[1], by providing insights for urban planning and traffic management to increase the efficiency of the transportation system [64]. Thus, traffic prediction has drawn great interests from government, traffic managers, and researchers for a long time.

The data used for traffic prediction mainly comes from two categories of sensors, static sensors (e.g., loop detectors and traffic cameras [20, 59]) and roving sensors (e.g., GPS device installed on a taxi and smartphones [36, 48, 66]). Data collected by static sensors is usually sampled at uniformly-spaced timestamps, while the data from roving sensors is temporally sparser and the amount of data varies from location to location. Due to high deployment and maintenance cost, the data from static sensors suffer from limited coverage as these sensors are typically installed at only limited/strategic locations like highway entrances and exits. On the other hand, the roving sensors cover larger areas, especially with the increasing ubiquity of smartphones. However, the nature of roving sensors results in more irregularity in data collection.

Given recent advances in machine learning, many data-driven approaches have been developed for traffic prediction by formulating it as a spatial-temporal prediction problem. The objective of these approaches is to model the traffic patterns simultaneously for multiple locations within a target region and then predict their traffic conditions in the future. A general framework of these approaches [4, 13, 36, 56, 63] is to integrate graph neural network (GNN) [6, 24] or graph convolutional network (GCN) [55] with long short-term memory (LSTM) [23]. Although these approaches have shown encouraging results in some isolated and clean testing scenarios, there are several major challenges that prevent them from reaching the same level of success in real-world traffic systems due to the nature of roving sensors:

Temporal sparsity: The data from roving sensors is often characterized by irregular/non-uniform temporal samples. For example, it is reported in previous work [48] that a large data set collected from tens of thousands of taxis has only 0.03% of all time intervals filled with data samples, even when the time interval was set to a coarse value of 30 minutes. For regions with only a few number of roving sensors, this issue can become even more severe. Such data can also significantly degrade the performance of standard temporal machine learning models such as LSTM. Although there have been many approaches on handling irregularly spaced time series, most of them use simple methods such as linear or polynomial interpolation without fully exploiting temporal correlations of the time series. Theses methods can have much worse performance given high sparsity rate.

Unevenly distributed: The amount of historical data from roving sensors may vary across different segments. As a result, we may have very limited data for certain regions, which makes it hard to train models for traffic prediction in these regions. Also, due to unevenly distributed data, information from neighbors of a location is often not available, which is not suitable to apply standard graph-based machine learning models such as GNN and GCN.

Spatial heterogeneity: Different roads and different regions can exhibit different traffic patterns given different environmental conditions (e.g., some regions may have more frequent road construction, and some roads are close to school district). Hence, models trained on one region cannot be directly used to predict traffic for roads within another region. This is one of the major reasons for many previous approaches [53, 54, 58] to divide a road network region into disjoint blocks and performing prediction separately for each block. However, given very limited training data for certain regions, it becomes challenging to build individual models for every region.

To address these challenges, we propose a new **Meta**-learning based **T**raffic **P**rediction framework, namely **MetaTP**, on temporally sparse and unevenly distributed road sensing data. Our framework consists of two major components, a time-series prediction model and a meta-knowledge learner. The time series prediction model is responsible for discovering temporal patterns from traffic data and uses learned patterns to interpolate values

[1] https://inrix.com/press-releases/2019-traffic-scorecard-us/

and conduct the prediction. The meta-knowledge learner aims to extract generalizable patterns across different regions by informing the initialization of the time-series prediction model.

In particular, the time-series prediction model uses a two-layer interpolation network (inspired by semi-parametric interpolation [1, 40, 41]), which transforms the input data onto predefined uniformly-spaced reference time points. This not only carries forward the information from individual time series, but also captures the cross-dimensional correlations of a multivariate time series. Through the interpolation network, the temporal sparsity issue of multivariate time series of traffic data can be well handled. After the interpolation, we use an LSTM model with two memory networks to conduct traffic prediction by leveraging the spatial and temporal patterns learned from the data.

The meta-knowledge learner is used to handle the problem of unevenly distributed data and the spatial heterogeneity. Our meta-knowledge learner follows the strategy of transferring useful knowledge amongst multiple regions. The intuition behind usability of knowledge transfer in our problem is that, despite the differences of data distributions and patterns in different regions, transferable knowledge in the form of shared common features exists and can assist in addressing the unevenly distributed data problem. Locations with sufficient data can be used to train a model, i.e. source regions, and the knowledge is then transferred to locations suffering from data scarcity, i.e., target regions. To achieve this goal, we employ a model-agnostic meta-learning (MAML) framework [18], which is capable of learning generalizable knowledge from source regions, and rapidly adapting to target regions with a handful of data examples via one or a couple of gradient descent steps. Specifically, MAML trains the base model in source regions to learn better initialization parameters, which are then used to adapt to target regions with a few data examples available.

Our main contributions can be summarized as follows:

- We propose a framework MetaTP to tackle the temporal sparsity and spatially unevenly distributed issues when using road sensing data for traffic prediction. In MetaTP, we use MAML to learn useful information from multiple source regions and quickly adapt the base model to target regions with limited data examples. To further enhance spatial-temporal correlations, we also add spatial and temporal memories to capture the global knowledge patterns.
- We implement a two-layer interpolation network to handle the temporal sparsity issue of irregularly spaced time series. In the interpolation process, the original information is carried over by univariate transformations, and the relationship among multiple time series is also captured.
- Extensive experiments have been carried out on two recently collected real-world datasets. We collect one dataset about traveling time on local road segments by deploying GPS tracking devices to 15 shuttles. The other dataset on highway traffic flow speed was collected by Caltrans Performance Measurement System (PeMS). The evaluation results show that our proposed MetaTP achieves the best performance under various settings. Ablation studies are conducted to demonstrate the effectiveness of major components of the framework.

The rest of the paper is organized as follows. In Section 2, we go over the related work. In Section 3, we introduce the system overview, and give out the formal problem definition. In Section 4, we describe MetaTP in details. In Section 5, we present the evaluations of MetaTP on two real world datasets. Discussion is provided in Section 6. Finally, we conclude the paper in Section 7.

2 RELATED WORK

In this section, we go over existing traffic prediction and knowledge transfer approaches, and discuss the comparison between the existing work and our approach.

2.1 Traffic Prediction

Traffic prediction has a long history in literature [12, 17, 33, 35, 51, 60]. Recent data-driven methods for traffic prediction can be divided into two categories, statistical methods and machine learning methods. Statistical methods, which are mainly based on autoregressive moving average (ARIMA) [42], Kalman filter [47], and Gaussian process [8, 52, 62], focus on modeling temporal dependencies. Conventional machine learning models such as linear regression [11] and support vector machine [46] have also been used for traffic prediction. [48] uses matrix decomposition to estimate the travel speed on each road segment. However, these approaches may not be suitable for modern traffic systems which commonly exhibit complex non-linear spatial and temporal patterns.

Recent work on traffic prediction mainly focuses on exploring spatial-temporal correlations using deep learning methods. For example, researchers have commonly used a combination of GNN (or GCN) with LSTM to capture spatial-temporal correlations [4, 13, 36, 56, 63, 65], and further enhanced the model via attention mechanism [16, 20]. The assumptions underlying these methods are that time series samples are evenly spaced across time, so that they can be fed into an LSTM; and location along with its neighbors are available simultaneously, so that it can aggregate information from its neighbors through GNN/GCN. However, data samples collected from roving sensors are often irregularly distributed across time, i.e., not all the variables in a multivariate time series are available at the same timestamp. LSTM is not directly applicable to this kind of data. Moreover, the frequency of different locations being sensed cannot be guaranteed, depending on routines of roving sensors. Therefore, a location may have different neighbors available at different timestamps, making GNN and GCN not directly applicable. Due to the sparsity and unevenly distributed nature of roving sensors, many approaches [54, 58] divide a city into disjoint large blocks and perform prediction on block level instead of road segments, since data in block level is distributed more evenly. However, how to effectively perform fine-grained prediction for individual road segments is an underexplored task in existing work.

The irregularly data distribution problem and the missing data problem are closely related, while there exist some differences. The missing data problem generally defines in the scenario that data points should be evenly recorded along time (e.g., every five minutes), but some values cannot be collected at certain time points, rising the missingness. The irregular data distribution problem typically occurs in continuous time space, and data points arrive irregularly without a predefined or "normal" frequency. Various imputation methods have been proposed to handle the data missing problem. To name a few, [5] imputes missing values by the linear combination of statistical estimates, [3] uses bi-directional recurrent neural network (RNN) combined with feature regression for missing value imputation, [43] explores local and global temporal dynamics to reconstruct missing values, and [32] employs generative adversarial learning to generate the missed data points in order to fool the discriminator. These missing value imputation methods are not directly applicable to the irregularly distributed traffic data. If discretizing an input time series into fixed-length intervals, the irregular distribution problem becomes a missing data problem, where empty intervals are said to carry missing values. However, this discretization is often considered as a preprocessing step outside of an imputation method. In contrast, we employ an interpolation method that takes the raw continuous time data as input, and learns the interpolation points in a end-to-end fashion, which is more flexible and avoids a separate step for assigning values to discrete time intervals.

2.2 Knowledge Transfer

As some locations may have much less sensor data than others, prediction for these locations relying on their own data may not be performed well. Transfer learning extracts and transfers previously learned knowledge to help in learning new tasks. Transfer learning approaches have been developed for sensor-based urban and traffic prediction such as spatial-temporal cross-domain transfer for cellular traffic prediction [57], semantically related multimodal feature transfer between cities [49], POI recommendation via user interest drift and transfer [21], calibration parameters transfer for air quality sensor [9], human mobility knowledge transfer [22] and adversarial social sensing transfer in transportation system [61]. Common transfer learning approaches include parameter sharing [39] and fine-tuning [21] strategies. Despite being able to transfer knowledge from source tasks to target tasks and mitigate the limited data issue, transfer learning cannot well handle the problem that target tasks only have a few training data available [18].

Meta-learning learns transferable knowledge from abundant training tasks, and then adapts to new tasks with a few examples. The learning process focuses on producing a model that is well generalizable to an unseen task. Regular transfer learning, in contrast, optimizes one or several training tasks and further fine-tunes the learned model parameters on testing tasks. Intuitively, transfer learning can well optimize training tasks, but often lacks the ability to generalize well on unseen target tasks with limited data. Domain adaptation [2, 44] learns to minimize the discrepancy between source and target distributions, so that knowledge learned from source domain can benefit the target one. Although unsupervised domain adaptation does not require label information in target domain, it still requires numbers of target data examples to learn the distribution of target domain, and thus has limited performance when only a few target examples are available.

Model-agnostic meta-learning (MAML) [18] learns initial parameters through training tasks, and quickly transfers them to a new task using one or a few gradient descent steps. Due to its model agnostic property, MAML has been applied to various deep learning models to enable knowledge transfer. MetaSense [19] applies meta-learning to handle the issue of countless individual conditions in human activity and speech recognition. MetaST [53] builds a spatial-temporal network and adapts it to a new city via meta-learning for traffic prediction. MetaST assumes that traffic volumes at all spatial locations in a city are recorded at each timestamp, and performs prediction of these locations jointly at a future timestamp. However, it cannot be directly applied in our scenario where the collected mobile sensory data is unevenly distributed: only limited locations in a city are monitored at each timestamp, and for certain locations, the data from spatial neighbors are rarely available. Besides, MetaST performs predictions on region block level, by dividing the city into rectangle blocks, whereas we try to solve a more fine-grained problem, i.e., traffic condition prediction on road segment level. Moreover, base prediction models are different. MetaST assumes that input data are complete and evenly distributed across time and space, while we develop a base model to handle the practical issue of data sparsity and irregularity by incorporating interpolation modules. Note that in [36, 37], the authors refer meta-knowledge as node and edge embeddings of a spatial graph learned from location attributes, which is essentially different from our method that meta-knowledge is the shared and transferable knowledge.

3 SYSTEM DESIGN

In this section, we first introduce the overview of our system architecture, and define some basic notations used in the paper. After that, we formally define the traffic prediction problem that the paper is trying to solve.

3.1 System Overview

We build a traffic monitoring system that collects traffic data from vehicles running in the road networks, transfers knowledge between regions, and performs quick prediction for target regions with a few observed data. The overall system framework is illustrated in Figure 1. We developed an Android application that continuously collects traffic information from various regions via smartphone's sensors equipped in different vehicles, e.g., bus and taxi, and on-board smartphones automatically upload the data to our server. In the server, the collected data pass through a series of preprocessing steps, including data cleaning, GPS map matching, etc. Due to the penetration rate of roving sensors, traffic data are inevitably sparse and irregular. To solve this problem, we propose a meta-learning based traffic prediction model, named MetaTP, that contains three main components: 1) interpolate the original data against a set of evenly spaced reference points; 2) capture temporal and spatial correlations from historical data, and train a prediction model for source regions ; 3) transfer knowledge from

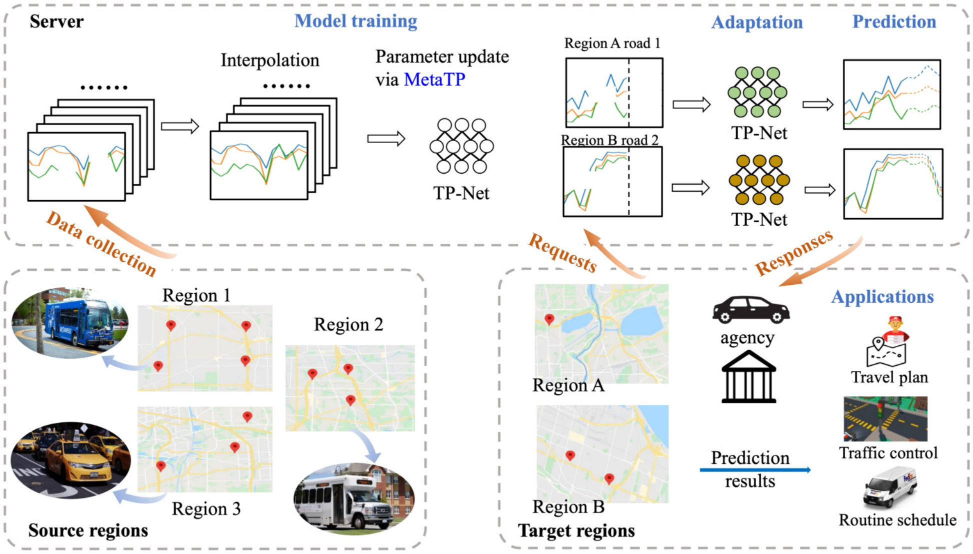


Fig. 1. Overview of system framework. Traffic data are collected from multiple source regions via sensors equipped in vehicles. The data are often sparse across time and space. We perform interpolation and knowledge transfer through the proposed MetaTP method, and obtain optimized parameters for the basic traffic prediction network TP-Net. In the target regions, when clients request future traffic conditions for certain roads, we can quickly adapt parameters in TP-Net with a few available data samples to account for the specific conditions, and perform prediction.

source to target regions, and perform prediction of future traffic for target regions. In particular, TP-Net is the base model that performs interpolation and prediction for each data sample. Through meta-learning, parameters in TP-Net are trained on a variety of traffic prediction tasks in source regions, and can extract patterns that are generalizable to other conditions. When traffic data from new regions are available, TP-Net can be quickly adapted to the new regions with a small amount of training data, and produce the prediction of future traffic conditions for the interested road segments. The predicted results can be utilized to help travel plan, routine schedule, traffic volume control, etc.

3.2 Basic Notations

The whole timeline of traffic data in a region can be divided into a series of consecutive but non-overlapping **time points**[2]. A **region** $\mathcal{V}$ is a geographical area (e.g., town, district or city) and $\{\mathcal{V}\}$ is a set of regions. We define **nodes** in a region as road segments, road intersections, and/or other spatial points whose traffic measurements are of our interest. **Traffic measurement** $\mathbf{x}_{n,t} \in \mathbb{R}^D$ for node n at time point t is a vector that includes D types of historical traffic information, e.g., traffic speeds (for multiple lanes if available), traffic volume, and average number of passengers per vehicle. For a node n, we collect a **multivariate time series** $\mathbf{x}_n = [\mathbf{x}_{n,1}; \mathbf{x}_{n,2}; ..., \mathbf{x}_{n,T}]^\top \in R^{T\times D}$, denoting traffic measurements across T time points. Note that in the rest of the paper, we use time point, timestamp and time interval interchangeably when it does not cause ambiguity, unless mentioned specifically.

[2] We use short time interval (e.g., every 5 minutes) as time point since average traffic information within an interval is often more meaningful than at a single timestamp.

3.3 Problem Definition

Previous work on traffic prediction often assumes the availability of traffic measurements for all interested nodes for a certain period of time, and such data can be readily used to train the predictive model. However, as we have mentioned in the introduction, this setting is not applicable in our problem using roving sensors due to the unevenly distributed data. In this paper, we focus on solving the problem that predicts future traffic measurements of any single road segment given its small amount of collected historical data. Thus, the traffic prediction problem in this paper can be defined as follows,

Given the collected traffic measurements for some nodes in a set of source regions $\{\mathcal{V}_{sr}\}$, our goal is to learn a robust model f_Θ for prediction, so that given a small amount of traffic data from node n in a target region $\mathcal{V}_{tg}$, we can quickly predict traffic data at future time points for the node, i.e.,

$$\widetilde{\mathbf{x}}_{n,t+1:t+\Delta t'} = \underset{\mathbf{x}_{n,t+1:t+\Delta t'}}{\arg\max}\, P(\mathbf{x}_{n,t+1:t+\Delta t'} | \mathbf{x}_{n,t-\Delta t+1:t}, f_\Theta), \tag{1}$$

where t denotes the current timestamp, Δt denotes the number of time points to consider the historical temporal information, $\mathbf{x}_{n,t-\Delta t+1:t}$ is node n's data from time point $t - \Delta t + 1$ to t, $\Delta t'$ is the number of future time points, i.e., length of prediction, P is the conditional probability, and $\widetilde{\mathbf{x}}_{n,t+1:t+\Delta t'}$ denotes the predicted values at the following $\Delta t'$ time points.

4 THE PROPOSED METATP MODEL

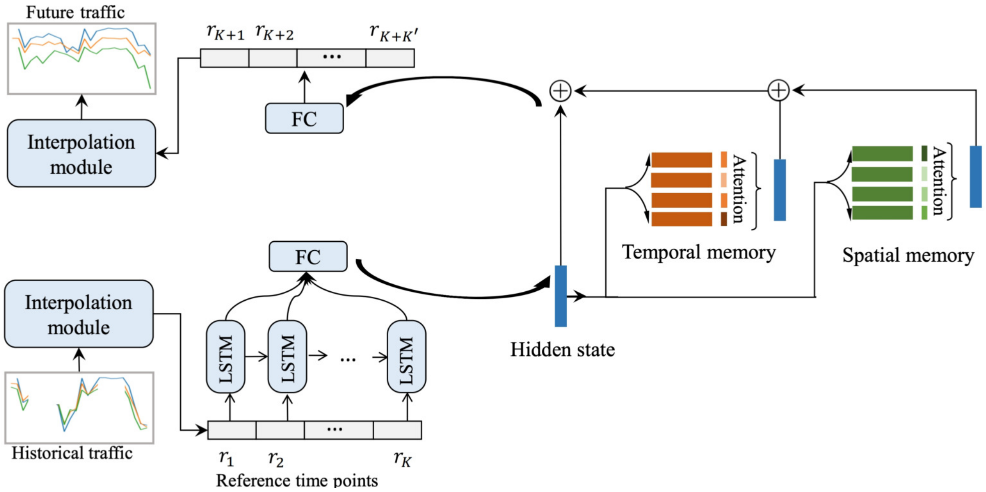


Fig. 2. Illustration of the proposed base model TP-Net. It includes two interpolation modules, the LSTM-based prediction module, and spatial and temporal memory networks. The first interpolation module maps the data samples onto one set of reference points which could have different granularity from the prediction series, whereas the second module maps the intermediate data to the reference points that are well aligned with prediction time series.

In this section, we provide details of the proposed framework. We first introduce the detailed components of the base model TP-Net, including generating evenly-distributed time series through interpolation, capturing global spatial and temporal correlations through memory networks, and performing prediction. After that, we provide details on parameter updates of the meta-knowledge learner, and give the overall learning algorithm.

4.1 Basic Prediction Model

The **T**ime Series **P**rediction **Net**work (**TP-Net**) is the base model which is shown in Figure 2. It mainly includes two interpolation modules, one LSTM layer, and spatial and temporal memory networks. The purpose of the interpolation module is to handle the data sparsity and irregularity issue of the roving sensors. In particular, the interpolation module fills in the "data gaps" by generating a new signal with regular time interval using the correlations amongst multiple dimensions of the original time series data. LSTM is then used to capture the temporal relationship among the interpolated data so as to generate the data representation. The obtained data representation (hidden state) is passed into the memory networks, which maintain the rich knowledge about the spatial and temporal information to assist with the time series prediction. The query results from the memories along with the hidden state are then passed through another interpolation module to map onto another set of reference points to be aligned with the prediction. We present details of the interpolation module, LSTM-based prediction module, and memory networks in the following subsections.

4.1.1 Generating Time Series Data with Regular Time Interval. The collected traffic data is often sparse and irregularly distributed due to limited number of roving sensors and the coverage rate of road segments. Figure 3(a) gives an example of data series with two dimensions (i.e., $D = 2$), representing traffic volume and speed. To tackle this problem, we implement an interpolation module inspired by [40] to transform the multivariate, sparse and irregular input series against a set of regularly distributed time points. Prediction module can be developed on top of the interpolated series. However, different from [40] which aims to produce a classification label using the interpolated series, our goal is to predict traffic values of original series at future time points of interest. Therefore, directly using an interpolation-prediction network as in [40] is infeasible in our problem. Our strategy is to further employ another interpolation module (whose details can be found in Section 4.1.2) to transform data from interpolated space to the original space, so that future values at interested time points can be estimated. Details on constructing an interpolation module are as follows.

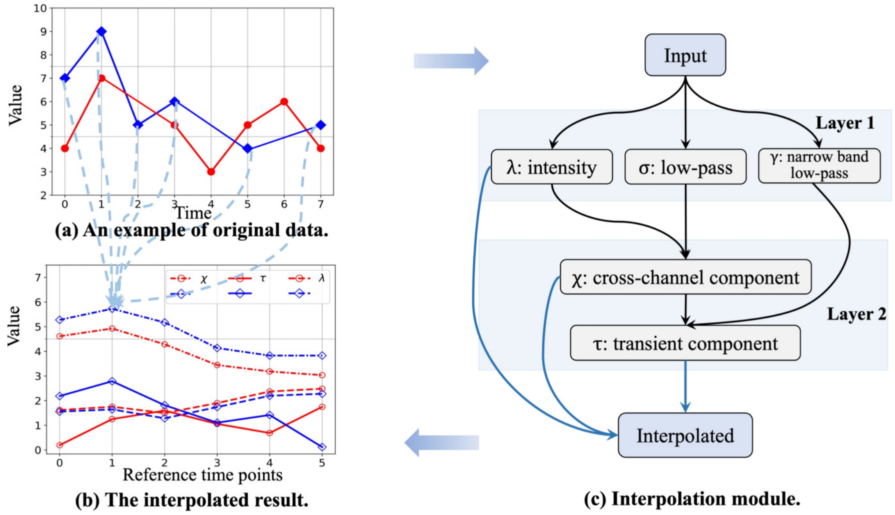
The intuition is to decompose each input series into multiple modes with each mode capturing certain type of information and then recover some modes by leveraging relationships with other input series in a collaborative way. In particular, the module consists of two layers. The first layer contains three transformations to extract different types of information from each time series. Then the second layer merges the information extracted from different time series, which allows the information from each input series to contribute to the interpolation of other time series. Finally, the interpolation module outputs multiple complete time series with regular time intervals. It is noteworthy that each output series may preserve different real-world interpretation to the original input data since it captures only certain aspects of traffic patterns learned from the original data. When we combine all the output series, their extracted patterns can be helpful for improving the traffic prediction.

Formally, for each of the D dimensions of the input $\mathbf{x}$ [3], the interpolation network produces a collection of interpolants that are defined at K **reference time points** $\mathbf{r} = [r_1, r_2, \dots, r_K]$. The interpolation module is illustrated in Figure 3. From the figure, we can see that each time series of the input (e.g., the blue or read line in Figure 3(a)) goes through a two-layer interpolation network and is transformed into three time series with reference time points as new timestamps (in Figure 3(b), the number of reference time points, i.e., K, is 6). The detailed implementations are given in the following paragraphs.

In the first layer, we perform three semi-parametric univariate transformations, i.e., intensity function $\boldsymbol{\lambda} \in \mathbb{R}^{K \times D}$, low-pass interpolation $\boldsymbol{\sigma} \in \mathbb{R}^{K \times D}$, and narrow band low-pass interpolation $\boldsymbol{\gamma} \in \mathbb{R}^{K \times D}$, on each of the D time series. Here we list the computation process of transformations for time series d at the reference time point r_k as an example. Firstly, for $\forall k \in \{1, \dots, K\}$ and $\forall d \in \{1, \dots, D\}$, the intensity function can be calculated as follows,

$$\lambda_{k,d} = F(r_k, \mathbf{t}_d, \alpha_d), \tag{2}$$

[3] In Section 4.1.1, for simplicity, we use $\mathbf{x}$ to represent $\mathbf{x}_{n,t-\Delta t+1:t}$.



(a) An example of original data.

(b) The interpolated result.

(c) Interpolation module.

Fig. 3. Interpolation Module. (a) shows a data sample with two dimensions. (b) depicts the interpolated result. For each dimension of input, there are three output dimensions, i.e., χ, τ and λ. (c) shows the structure of the interpolation module.

where $\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_D\} \in \mathbb{R}^D$ is a hyperparameter to control the penalty of time differences, $\mathbf{t}_d \in \mathbb{R}^{\Delta t}$ is a list of time points for time series d of the input, and $F(r, \mathbf{t}, \alpha) = \sum_{t \in \mathbf{t}} \exp\left(-\alpha(r-t)^2\right)$ indicates how close the time points of the input are to a reference time point r. Input series having closer time points with references yields larger intensity function. Secondly, the low pass transformation can be represented as,

$$\sigma_{k,d} = \frac{\exp\left(-\alpha_d (r_k - \mathbf{t}_d)^2\right)^\top \mathbf{x}_d}{F(r_k, \mathbf{t}_d, \alpha_d)}, \tag{3}$$

where $\mathbf{x}_d$ is the d-th dimension of the input $\mathbf{x}$. This low pass transformation means that if a value from input is closer to a certain reference point r_k, it will have more impact on the interpolated value at r_k. Finally, the narrow band low pass interpolation is represented as,

$$\gamma_{k,d} = \frac{\exp\left(-\kappa\alpha_d (r_k - \mathbf{t}_d)^2\right)^\top \mathbf{x}_d}{F(r_k, \mathbf{t}_d, \kappa\alpha_d)}, \tag{4}$$

where $\kappa > 1$ is a hyperparameter. This adds exponential penalty to values in the input that are far away from reference points so that their impact on the interpolated results will be reduced.

The second interpolation layer integrates information for each reference point from the multivariate series. First, a cross-dimensional interpolation $\boldsymbol{\chi} \in \mathbb{R}^{K \times D}$ is calculated as follows,

$$\chi_{k,d} = \frac{\sum_{d'} \rho_{d,d'} \lambda_{k,d'} \sigma_{k,d'}}{\sum_{d'} \lambda_{k,d'}} = \frac{\sum_{d'} \rho_{d,d'} \exp\left(-\alpha_d (r_k - \mathbf{t}_d)^2\right)^\top \mathbf{x}_{d'}}{\sum_{d'} \lambda_{k,d'}}, \tag{5}$$

where $\boldsymbol{\rho} \in \mathbb{R}^{D \times D}$ is learnable hyperparameter that indicates the correlations among time series. Since we are dealing with multivariant time series, there exists correlations among different dimensions. Eq. (5) is designed to capture these cross-dimension relationships. It calculates the weighted sum across all input dimensions on every output reference point. More specifically, the value of the k-th reference point of the d-th dimension of the output, i.e., $\chi_{k,d}$, has contributions from the input of all dimensions. The learnable parameter ρ and the radial basis function controls the relationship. Then, a transient component $\boldsymbol{\tau} \in \mathbb{R}^{K \times D}$ is defined as the difference between the narrow band low-pass interpolation $\boldsymbol{\gamma}$ and the cross-dimension interpolation $\boldsymbol{\chi}$, i.e.,

$$\boldsymbol{\tau} = \boldsymbol{\gamma} - \boldsymbol{\chi}. \tag{6}$$

Finally, we concatenate $\boldsymbol{\lambda}$, $\boldsymbol{\chi}$ and $\boldsymbol{\tau}$ to get the interpolated result,

$$\mathbf{y}_{n,r_1:r_K} = Interpolate_1(\mathbf{x}_{n,t-\Delta t+1:t}) = [\boldsymbol{\lambda}; \boldsymbol{\chi}; \boldsymbol{\tau}], \tag{7}$$

where $\mathbf{y}_{n,r_1:r_K} \in \mathbb{R}^{K \times 3D}$. In this way, we obtain the interpolated data $\mathbf{y}_{n,r_1:r_K}$ at reference time points for a multivariate time series $\mathbf{x}_{n,t-\Delta t+1:t}$ collected from node n.

4.1.2 Temporal Prediction. Once obtaining complete data series that cover all the reference time points, we can feed such data series into temporal analysis tools to conduct the prediction. Recurrent neural network (RNN) is one of the most widely used methods given its ability to extract non-linear temporal relationships through neural network layers. Variants of RNN, e.g., long short-term memory (LSTM) [23], and gated recurrent units (GRUs) [10], have found a lot of success in the traffic prediction using complete training data series [4, 13, 36, 53, 56, 63]. Without loss of generality, here we use LSTM in our prediction model and all baseline models. An LSTM cell can be formulated for interpolated time series $\mathbf{y}_{n,r_1:r_K}$ as follows,

$$\begin{aligned}
\mathbf{f}_{n,r_k} &= \sigma(\mathbf{W}_f \mathbf{y}_{n,r_k} + \mathbf{U}_f \mathbf{h}_{n,r_{k-1}} + \mathbf{b}_f), \\
\mathbf{i}_{n,r_k} &= \sigma(\mathbf{W}_i \mathbf{y}_{n,r_k} + \mathbf{U}_i \mathbf{h}_{n,r_{k-1}} + \mathbf{b}_i), \\
\mathbf{o}_{n,r_k} &= \sigma(\mathbf{W}_o \mathbf{y}_{n,r_k} + \mathbf{U}_o \mathbf{h}_{n,r_{k-1}} + \mathbf{b}_o), \\
\tilde{\mathbf{c}}_{n,r_k} &= \tanh(\mathbf{W}_c \mathbf{y}_{n,r_k} + \mathbf{U}_c \mathbf{h}_{n,r_{k-1}} + \mathbf{b}_c), \\
\mathbf{c}_{n,r_k} &= \mathbf{f}_{n,r_k} \odot \mathbf{c}_{n,r_{k-1}} + \mathbf{i}_{n,r_k} \odot \tilde{\mathbf{c}}_{n,r_k}, \\
\mathbf{h}_{n,r_k} &= \mathbf{o}_{n,r_k} \odot \tanh(\mathbf{c}_{n,r_k}),
\end{aligned}$$

where $\mathbf{f}$, $\mathbf{i}$, and $\mathbf{o}$ are forget, input, and output gates' activation vectors, respectively, $\odot$ stands for Hadamard product, and $\mathbf{W}_*$, $\mathbf{U}_*$ and $\mathbf{b}_*$ ($* \in \{f, i, o, c\}$) are learnable parameters. $\mathbf{h}_{n,r_1:r_K}$ is the temporal representation of the sample. The forget gate is used to filter the information inherited from the previous time stamp, and the input gate layer is used to filter the candidate cell state at the current time. The output gate is introduce to future filter the obtained cell state to generate hidden representation.

As is shown in Figure 2, we pass the representation through a fully connected (FC) network to get the intermediate prediction,

$$\hat{\mathbf{y}}_{n,r_{K+1}:r_{K+K'}} = FC(\mathbf{h}_{n,r_1}, \dots, \mathbf{h}_{n,r_K}), \tag{8}$$

where K' is the number of reference points of the intermediate prediction.

However, due to the interpolation, the time difference, i.e., interval size, between two consecutive time points of this intermediate prediction is no longer the same as the original data nor the prediction, since the number of reference points K may not be equal to the input length Δt. Therefore, we propose to add another interpolation module, as shown in Figure 2, to map the intermediate prediction back to the original time points. The reference points of this interpolation module are the same with time points of the prediction. The module can be represented as follows,

$$\mathbf{x}'_{n,t+1:t+\Delta t'} = Interpolate_2(\hat{\mathbf{y}}_{n,r_{K+1}:r_{K+K'}}). \tag{9}$$

Since the interpolation module transforms each time series into three series, after the second interpolation, we use another FC to get the estimated results,

$$\widetilde{\mathbf{x}}_{n,t+1:t+\Delta t'} = FC(\mathbf{x}'_{n,t+1:t+\Delta t'}). \tag{10}$$

We use the mean squared error (MSE) loss function to measure the difference between the estimated value and the ground truth as follows,

$$\mathcal{L} = \sum_{n} \sum_{t} (\widetilde{\mathbf{x}}_{n,t+1:t+\Delta t'} - \mathbf{x}_{n,t+1:t+\Delta t'})^2. \tag{11}$$

4.1.3 Spatial and Temporal Memory Networks. Here we aim to further leverage the global spatial and temporal patterns shared by different regions and time periods to improve the traffic prediction in a collaborative fashion. For example, traffic flow around a school and a company may have similar daily periodic patterns, while these patterns are different from traffic flow around a state park. Hence, we propose to use the spatial memory [50, 53] to aggregate the information from similar locations and use such aggregated information to augment the prediction model in representing individual locations. Moreover, traffic often shows certain temporal patterns, e.g., morning rush hour for the same location almost always happens around the similar time period of each day. Therefore, in addition to the spatial memory, we also implement a temporal memory network. Two memory networks together can capture more complicated global spatial-temporal relationships.

Although memory network has been used in [53], our memory network module are different in two aspects. First, [53] performs clustering on locations using historical averaged traffic data of each location, so that data samples from the same cluster are assigned with the same cluster label when querying the memory. However, the lookback length of historical data could affect the clustering result, e.g., historical patterns within a week and a month could be very different, making it hard to decide proper cluster labels. In contrast, we use the external geographic knowledge of road networks, e.g., number of lanes, and points of interests to generate clusters for spatial memory, which can be more stable. Second, only clustering locations in [53] may not be sufficient to capture the global temporal relationship among data samples, because traffic conditions at multiple consecutive time points within the same time period, e.g., morning rush hour, often share some common characteristics. By using only location clustering, two samples from the same clustering with different timestamps will be treated equally when querying the memory. Our utilization of both spatial and temporal memory networks resolves this issue and enables more complex patterns to be captured.

Spatial Memory. We first split all nodes into M clusters using k-means clustering algorithm based on the collected node information, e.g., number of lanes, speed limit, and road geography information. Then the spatial memory can be viewed as a $M \times C$ matrix, denoted as $\mathcal{M}_s$, where each row stands for the representation of one type of summary for historical data, i.e., the memory for the data from one certain spatial cluster; and C stands for the memory dimension to store the information for each type of spatial knowledge.

The output of the second interpolation $\mathbf{x}'_{n,t+1:t+\Delta t'}$ from Eq. (9) first goes through a fully connected layer, i.e.,

$$\mathbf{q}^S_{n,t} = FC(\mathbf{x}'_{n,t+1:t+\Delta t'}), \tag{12}$$

in order to get the information needed to interact with the memory. Then, we use attention mechanism [45] to query the memory to get spatial knowledge. In other words, we calculate the production between $\mathbf{q}^S_{n,t}$ and each row of the memory, which can be viewed as the similarity between the input and the existing spatial memory knowledge. Then these similarities are normalized by a softmax function to get the final attention score. The calculation of the attention score with the m-th memory row is shown as follows,

$$\mathbf{p}^S_{n,t}(m) \triangleq \frac{\exp\left(\mathbf{q}^S_{n,t} \cdot \mathcal{M}_s(m)\right)}{\sum_{m' \in M} \exp\left(\mathbf{q}^S_{n,t} \cdot \mathcal{M}_s(m')\right)}, \tag{13}$$

where $\cdot$ is the dot product operation and $\mathcal{M}_s(m)$ is the m-th row in $\mathcal{M}_s$. A higher attention score indicates that the current sample is more likely to be similar to historical samples that this certain memory row represents.

Once we have the attention score, we can get the final spatial information that this sample can pull from the historical data by computing the weighted sum of the attention score with each memory row. This computation process can be expressed using the following equation,

$$\mathbf{z}_{n,t}^{S} = \sum_{m=1}^{M} \mathbf{p}_{n,t}^{S}(m)\mathcal{M}_{s}(m). \tag{14}$$

After we have processed the input, we need to update the memory to reflect the new knowledge that is learned from the sample so as to serve further queries better. In order to update the memory to get an evolving and better representation, we can calculate the loss for the current query. Specifically, since we have already clustered each node, the sample from a certain cluster is expected to have stronger correlation with samples from the same cluster. In other word, the attention score between the sample's representation with the corresponding memory row should be the highest. Therefore, we formulate the loss of the spatial memory as follows,

$$\mathcal{L}_{n,t}^{S} = -\log(\mathbf{p}_{n,t}^{S}) \cdot \mathrm{S}(n), \tag{15}$$

where $\mathrm{S}(n)$ is the one-hot encoding for the clustering that the sample's node n belongs to.

Temporal Memory. The pattern of a sample from the morning rush hour is often more similar to another one from a rush hour than a sample from the late night. Therefore, we group time points into non-overlapping clusters so that extra knowledge for new input data can be learned from historical data falling into similar temporal clusters. When the number of clusters is given, the criteria to divide the timeline can be based on the minmax strategy, where the difference of inter-cluster is minimized, and the difference of intra-cluster is maximized. In this paper, for simplicity, we generate the clusters based on the traffic measurements in different period of each day, i.e., morning rush hour, midday, afternoon rush hour, and evening. We will leave the study of optimal temporal grouping in future work, including how to determine the number of groups. For temporal memory network $\mathcal{M}_t \in \mathbb{R}^{M' \times C'}$, the attention score $\mathbf{p}_{n,t}^{T}$, the pulled information from temporal memory $\mathbf{z}_{n,t}^{T}$, and the loss for temporal memory $\mathcal{L}_{n,t}^{T}$ can be obtained in the similar way with the spatial memory, so that we omit the detailed computation process.

Concatenate $\mathbf{x}'_{n,t+1:t+\Delta t'}$ with the pulled information from both memories to get the enhanced representation,

$$[\mathbf{x}'_{n,t+1:t+\Delta t'}; \mathbf{z}_{n,t}^{S}; \mathbf{z}_{n,t}^{T}], \tag{16}$$

which is used to generate the final prediction by Eq. (10). And the overall loss function can be finalized as follows,

$$\mathcal{L}_a = \mathcal{L} + \sum_{n}\sum_{t}\left(\eta_1 \mathcal{L}_{n,t}^{S} + \eta_2 \mathcal{L}_{n,t}^{T}\right), \tag{17}$$

where $\mathcal{L}$ is the prediction loss defined in Eq. (11), and η_1 and η_2 are two hyperparameters that control the loss weights of spatial and temporal memory, respectively.

4.2 Meta-Knowledge Learner

Despite the capability of handling irregularly distributed time series of TP-Net, its performance is restricted with limited available data in target regions. Since different regions could have quite different sensor coverage, directly applying the learned model from source regions to target regions with a fine-tune step, cannot robustly capture the diverse spatial and temporal characteristics of each region, and may cause unstable results or even the risk of negative transfer. To enable robust knowledge transfer, we train the parameters of TP-Net via model-agnostic meta-learning (MAML) [18]. The underlying idea of MAML is that there exist initial parameters that are transferable among tasks, so that the initial parameters can be adapted to a new task quickly with a few data, and produce robust generalization performance.

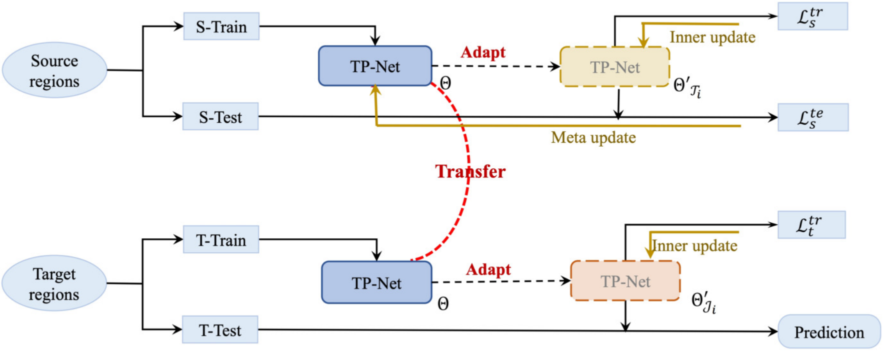


Fig. 4. Schematic diagram of the knowledge transfer module. In the meta-knowledge transferring process, abundant traffic prediction tasks from source regions are used to optimize initial parameters Θ of TP-Net. Each task $\mathcal{T}_i$ from source regions contains two disjoint sets of data examples, i.e., training and test sets, denoted as S-Train and S-Test. S-Train set is used to update the initial parameters Θ to task-dependent parameters $\Theta'_{\mathcal{T}_i}$, which simulates adapting parameters to the specific condition of a task; and S-Test is used to evaluate the performance of $\Theta'_{\mathcal{T}_i}$ on test set, and update initial parameters Θ based on the acquired loss on S-Test. In target regions, with a few observed traffic data samples, the optimized parameters Θ of TP-Net can be quickly adapted to the new condition, and provide accurate prediction results for the given traffic data.

4.2.1 Parameter Updates. For simplicity, we use Θ to denote all parameters from TP-Net except memory networks, and represent TP-Net as f_Θ. For each source region, we randomly sample N_t tasks $\{\mathcal{T}_1, \ldots, \mathcal{T}_{N_t}\}$. Each task $\mathcal{T}_i$ contains two disjoint sets of data samples, i.e., training set $\mathcal{D}_{\mathcal{T}_i}^{tr}$ (denoted as S-Train in Figure 4), and test set $\mathcal{D}_{\mathcal{T}_i}^{te}$ (denoted as S-Test in Figure 4). We ensure the training set and test set have no overlapping data. Training sets are used for learning a task-specific parameter $\Theta'_{\mathcal{T}_i}$ for each task $\mathcal{T}_i$ through a few gradient descent updates, and test sets are used for evaluating the generalization of the task-specific parameters and updating initial parameters Θ. We use $\mathcal{L}_s$ to denote the prediction loss for source regions, and it is calculated by Eq. (11). The task-specific parameters $\Theta'_{\mathcal{T}_i}$ can be obtained using $\mathcal{D}_{\mathcal{T}_i}^{tr}$ with gradient descent as,

$$\Theta'_{\mathcal{T}_i} = \Theta - \beta \nabla_\Theta \mathcal{L}_{s,\mathcal{T}_i}^{tr}(\Theta, \mathcal{D}_{\mathcal{T}_i}^{tr}), \tag{18}$$

where β is a predefined learning rate. Eq. (18) shows one step of gradient descent, and in practice, we use several gradient descent steps (e.g., 5 steps) to update from Θ to $\Theta'_{\mathcal{T}_i}$. This process of adapting and updating parameters for a specific task is illustrated as an inner loop arrow in Figure 4.

With task-specific parameters $\Theta'_{\mathcal{T}_i}$, we train the model parameters Θ by optimizing the performance of $f_{\Theta'_{\mathcal{T}_i}}$ on test set $\mathcal{D}_{\mathcal{T}_i}^{te}$. Additionally, memory networks $\mathcal{M}_s$ and $\mathcal{M}_t$ are updated together with Θ in the meta update process, instead of task-specific adaptation, since memory networks capture global spatial-temporal information and are shared across tasks. The meta-objective function is defined as,

$$\min_{\Theta, \mathcal{M}_s, \mathcal{M}_t} \sum_{\mathcal{T}_i} \mathcal{L}_{s,\mathcal{T}_i}^{te}(\Theta'_{\mathcal{T}_i}, \mathcal{D}_{\mathcal{T}_i}^{te}) + \sum_n \sum_t (\eta_1 \mathcal{L}_{n,t}^S + \eta_2 \mathcal{L}_{n,t}^T), \tag{19}$$

where $\Theta'_{\mathcal{T}_i}$ is obtained by Eq. (18), and the second term denotes the memory losses of test set $\mathcal{D}_{\mathcal{T}_i}^{te}$ in $\mathcal{T}_i$. During meta-optimization, the gradient is computed with respect to Θ, $\mathcal{M}_s$ and $\mathcal{M}_t$, while the test loss $\mathcal{L}_{s,\mathcal{T}_i}^{te}$ is computed with respect to task-specific parameters $\Theta'_{\mathcal{T}_i}$ and evaluated with data $\mathcal{D}^{te}_{\mathcal{T}_i}$. We illustrate this update step using the meta update arrow in Figure 4.

4.2.2 Target Adaptation. After TP-Net is trained using source regions, it can be adapted to various target regions with a few collected data in response to traffic prediction requests. Suppose that we have a few historical traffic measurements from i-th node in a target region, and the client wants to know future traffic condition of this node, which forms a task $\mathcal{J}_i$, we can calculate the model parameters for $\mathcal{J}_i$ as,

$$\Theta'_{\mathcal{J}_i} = \Theta - \beta \nabla_{\Theta} \mathcal{L}^{tr}_{t,\mathcal{J}_i}(\Theta, \mathcal{D}^{tr}_{\mathcal{J}_i}), \tag{20}$$

where $\mathcal{D}^{tr}_{\mathcal{J}_i}$ is the training set of the i-th node in the target region, and it contains a few historical time series; $\mathcal{L}_{t,\mathcal{J}_i}$ is the loss calculated using $\mathcal{D}^{tr}_{\mathcal{J}_i}$. For a time series $\mathbf{x}_{i,t-\Delta t+1:t}$ in testing set $\mathcal{D}^{te}_{\mathcal{J}_i}$, we can apply TP-Net with parameters $\Theta'_{\mathcal{J}_i}$ to obtain estimated future traffic measurements $\widetilde{x}_{i,t:t+\Delta t'}$ as,

$$\widetilde{\mathbf{x}}_{i,t:t+\Delta t'} = f_{\Theta'_{\mathcal{J}_i}}(\mathbf{x}_{i,t-\Delta t+1:t}). \tag{21}$$

Prediction requests from other nodes in target regions can be fulfilled through Eq. (20) and Eq. (21) similarly as for node i.

Through the meta-learning process, TP-Net has experienced various tasks from source regions via training steps in Section 4.2.1, and it can be fast and effectively adapted to the target condition in Section 4.2.2 with the learned meta-knowledge. The overall framework of MetaTP is shown in Algorithm 1.

Algorithm 1: Overall Algorithm of MetaTP

```
Input: Dataset; Stepsize of optimizers: $\beta$; Weights of memories' loss: $\eta_1, \eta_2$
Result: Predicted traffic information at future timestamps
Randomly initialize parameters $\Theta$ in TP-Net, spatial memory $\mathcal{M}_s$, and temporal memory $\mathcal{M}_t$
Generate spatial and temporal clusters for source nodes and time points respectively
while not done do
    Randomly sample a set of regions
    Randomly sample a batch of nodes from each selected region
    for each node do
        Sample $\mathcal{D}^{tr}_{\mathcal{T}_i}$ and $\mathcal{D}^{te}_{\mathcal{T}_i}$
        Calculate the loss of $\mathcal{D}^{tr}_{\mathcal{T}_i}$ by Eq. (17)
        Update $\Theta'_{\mathcal{T}_i}$ using gradient descent by Eq. (18)
        Evaluate loss $\sum_{\mathcal{T}_i} \mathcal{L}_{s,\mathcal{T}_i}(\Theta'_{\mathcal{T}_i}, \mathcal{D}^{te}_{\mathcal{T}_i})$ on $\mathcal{D}^{te}_{\mathcal{T}_i}$
    end
    Update $\Theta$, $\mathcal{M}_s$ and $\mathcal{M}_t$ by Eq. (19)
end
/*Target Adaptation*/
for each node in target regions do
    Sample $\mathcal{D}^{tr}_{\mathcal{J}_i}$ and $\mathcal{D}^{te}_{\mathcal{J}_i}$
    Adapt $\Theta$ to $\Theta'_{\mathcal{J}_i}$ on $\mathcal{D}^{tr}_{\mathcal{J}_i}$ with a few gradient descent steps by Eq. (20)
    Get the prediction for $\mathcal{D}^{te}_{\mathcal{J}_i}$ based on $\Theta'_{\mathcal{J}_i}$ by Eq. (21).
end
```

5 EVALUATION

In this section, we evaluate the performance of MetaTP on two traffic prediction tasks, i.e., one is for local travel time prediction and the other is for highway traffic speed prediction.

5.1 Experimental Settings

In this section, we first introduce the measurement metrics, and then briefly describe the baseline methods to compare with. At the end, we give out the choices of hyperparameters and experimental details.

Metrics. We use the mean absolute error (MAE) and the rooted mean square error (RMSE) to evaluate the performance,

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|, \quad \text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}, \tag{22}$$

where N is the number of instances, $\hat{y}_i$ is the prediction result and y_i is the ground truth.

Baseline methods used in this paper can be grouped into two categories, i.e., non-transfer baselines and transfer based baselines.

Non-transfer Baselines:

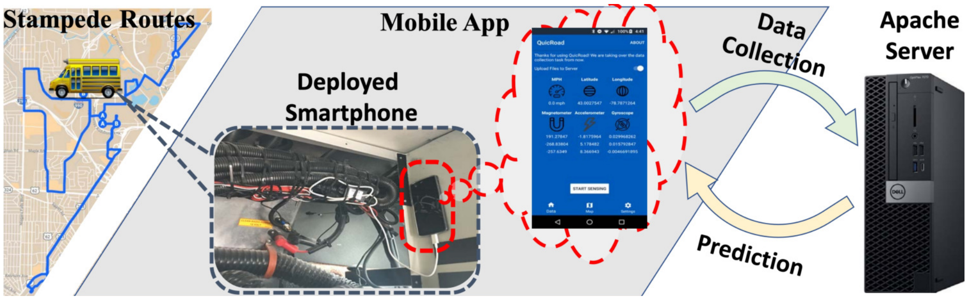
- **Historical Average (HA)**: Since traffic data of the near future has close relationship with the past timestamps, following [20, 27], we calculate the average traffic information for each time series, and use it as the predicted value for future timestamps.
- **Vector Autoregression (VAR)**: VAR is a traditional multivariate time series analysis model. It considers temporal relationships and dependencies among variates.
- **Linear Regression (LR)**: LR is a commonly used method for time series predictive analysis. Here a variable is considered as missing at a timestamp, if other variables are available. We first fill in the "missing" values with the forward filling method, i.e., use the last observed value to replace the missing. For each time series, we use past data points to fit an LR model, and then conduct prediction.
- **Gaussian Processes (GP)** [52]: GP based regression model is a kernel-based machine learning method that can be used to solve the traffic prediction problem.

Transfer Baselines:

- **TransFT**: Transfer learning with fine-tuning uses LSTM with forward filling to train on the source data. For the target tasks, it uses a few training samples in target domain to fine-tune the model, and then applies the tuned model on test data in target.
- **MAML-M**: It is based on MAML [18], which aims to learn a better initialization for target tasks from multiple source tasks. The base model is LSTM with missing values filled with mean of observed values.
- **MAML-F**: It is also based on MAML. The difference between MAML-F and MAML-M is that we use forward filling on LSTM.
- **MetaST** [53]: MetaST builds a spatial-temporal network that combines CNN and LSTM, and adapts it to a new city via meta-learning. In [53], each city is divided into non-overlapping rectangle blocks and the prediction is done on the block level. Therefore, it uses CNN to capture spatial relationships in the Euclidean space and aggregate predictions in rectangular blocks. Whereas in this paper, we deal with traffic prediction on the road segment level, where the road network forms non-Euclidean and irregular graphs, which is not suitable to apply CNN. Thus, we remove the CNN in the implementation. Missing values are replaced with forward filling.
- **MetaTP-Lite**: To illustrate the effectiveness of memory networks, we also implement a simplified version of the proposed framework without adding memory networks. Temporal missingness (i.e., sparsity) is handled via interpolation proposed in Section 4.1.1.

Implementation Details: All transfer based baselines are implemented using PyTorch 1.15[4] [38] with Python 3.7. MSE is chosen as the loss function. Adam optimizer [25] is used as the optimization method with gradient clipping. Experiments are tested with Nvidia Titan X Pascal. Different choices of hidden size in the range from 16 to 1024 has been tried out. Best performance is achieved at 128 which is used for all methods that utilize the LSTM module. The size of spatial and temporal attention memories are set to be 8.

5.2 Travel Time Prediction on Stampede Dataset



Thanks for using QuicRoad! We are taking over the data collection task from now.

Upload Files to Server

MPH	Latitude	Longitude
0.0 mph	43.0027547	-78.7871264

Magnetometer	Accelerometer	Gyroscope
191.27847	-1.8175964	0.029968262
-268.83804	5.178482	0.015792847
-257.6349	8.366943	-0.0046691895

Fig. 5. Demonstration of the data collection system of Stampede. From left to right, the sub-figures are Stampede routes, a deployed smartphone in the overhead bin of a Stampede shuttle, a screenshot of the data collection application, and an Apache server that stores and processes collected data.

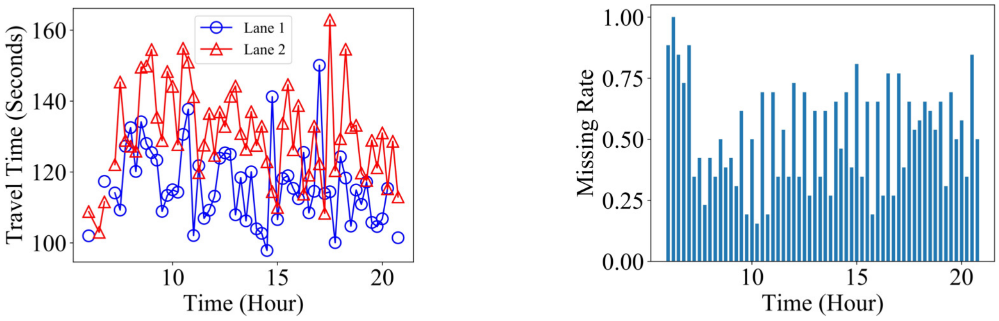
5.2.1 *Stampede Dataset.*

In order to study the traffic prediction in local road network, we have deployed GPS tracking devices (i.e., Android smartphones with our developed GPS tracking application) on 15 shuttles named **Stampede**, which run among different university campuses and between campuses and supermarkets. We got in touch with the local company that operates Stampede shuttles via university transportation authorization on the potential usage of the collected data (e.g., travel time estimation, and driving behavior monitoring) and got permission to deploy the sensors. For each shuttle station, the scheduled service interval between shuttles during daytime of weekdays is as few as 5 minutes, while during night or weekends it can be as long as 60 minutes. The sampling rate of GPS is about 1Hz. The collected GPS data are uploaded to our Apache web server whenever the smartphone is connected to WiFi. There are WiFi access points across the campus that are tested to be capable to upload all collected data. The Stampede routines, an example of the deployment of a tracking device in the overhead bin of the Stampede shuttle, the screen shot of the Android application, and the Apache web server are depicted in Figure 5.

In this evaluation, we use Stampede data collected from February 1, 2020 to February 29, 2020. The routes have been split into 14 road segments. Each road segment (i.e., node) contains the running time information for both directions. The timeline is divided into 15 minutes intervals. There are totally 24,018 valid records. From our observation, the running times for each road segment during late night (after 9pm) and early morning (before 6am) are very stable during that period, so we choose data collected between 6am and 9pm to run experiments. There are 21,305 records within this time period. We split data into training and testing by ratio 8:2. Road network information, including the number of lanes per direction, number of stop signs, number of traffic lights, speed limits, and the GPS location of the center point of each road segment, has been used to cluster road segments into 4 different groups, based on the k-means clustering algorithm. Temporal clustering is done by cutting each day into 8 non-overlapping clusters since nearby time points usually have closer relationship. More information,

[4] https://pytorch.org/

however, can be added, like weekday, weekends, holiday, etc, to get a possible better clustering result which we leave for future work to study.



(a) Travel Time

(b) Missing Rate

Fig. 6. Average travel time and missing rate for Stampede dataset on a selected road segment in different time of the day.

Figure 6a and Figure 6b show the average travel time and the average missing rate, during each interval for one selected road segment, respectively. Here, the missing rate is defined as, for one input data, the ratio between the number of timestamps without traffic data and the data length. Note that our input data is in continuous time space. The missing rate here is used to quantify how sparsity the collected data is. We denote the average missing rate in time series by p. From Figure 6a, we can see that the two time series of travel time corresponding to two directions have evidently correlation. Both directions have relative higher travel time during the morning rush hour and the evening rush hour. The missing rate reflects the issue of data sparsity. Figure 6b shows that missing rates of consecutive time intervals can be different because not all shuttles are equipped with GPS tracking devices. This indicates the unevenly-distributed nature of roving sensory data. As this road is on the routine of a round trip between two campuses and it takes about half an hour for one way trip, for three consecutive intervals, the first and third intervals have closer missing rates. We want to mention that the missing rate here does not impact the accuracy of the collected data. The GPS receiver on smartphones we use can achieve an accuracy of a few meters.

5.2.2 Performance w.r.t. Missing Rate p. Since the temporal sparsity situation may vary from one case to another, in this section, we study how well MetaTP performs with different missing rate p. For p's range from 10% to 90%, we run all methods on Stampede dataset under settings that the length of each source data is 2 hours (i.e., Δt=8), and prediction length is 45 minutes (i.e., $\Delta t'$=3). Results are shown in Table 1. From the table we can see that for each method, with p increasing, the performance generally gets worse. MetaTP outperforms all the baselines for all choices of p. The reason is that MetaTP can alleviate the impact of temporal sparsity not only by learning the inner relationships among different time points, but also by learning the cross-dimensional correlations among time series. The performance of MetaTP keeps strong even when p becomes very high. MetaTP-Lite performs slightly worse than MetaTP which illustrates that the memory networks can help enrich the data representation with learned spatial and temporal knowledge from historical data. But it still outperforms MAML-M and MAML-F, which indicates the effectiveness of the interpolation network.

HA works well since the traffic usually won't change much in a short period of time. But due to the lack of more sophisticated analysis, the performance gap between HA and MetaTP enlarges when the prediction length p increases. When p gets larger, the performance of VAR deteriorates quickly, especially in terms of RMSE.

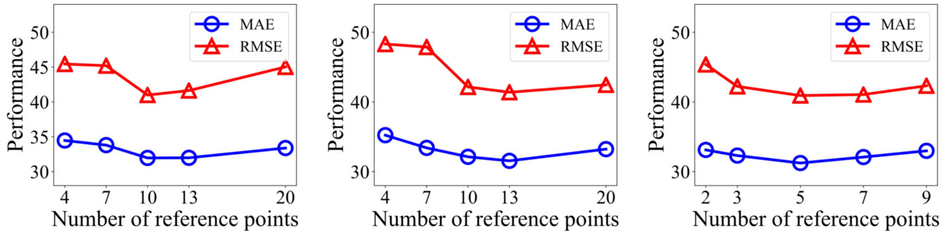
Table 1. Performance on Stampede dataset w.r.t. missing rate and prediction length.

	Methods	10%		30%		50%		70%		90%	
		MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
Missing rate	HA	30.2963	36.2805	33.4543	45.8264	35.1960	48.4916	35.8237	49.3912	41.6951	57.1164
	VAR	30.0668	36.2997	33.8191	46.5974	34.6367	47.0917	35.3368	47.3535	42.1719	58.4321
	LR	37.5307	47.5624	47.5060	63.7812	50.8413	67.5694	51.2780	69.9089	41.7913	57.4964
	GP	30.0348	35.9824	33.8330	46.2711	36.0585	49.8693	36.7091	50.0090	41.7061	57.0703
	MAML-M	31.4843	38.0773	32.0669	43.7039	33.1243	44.3655	33.7721	45.4455	33.8768	46.6542
	MAML-F	31.3826	37.8950	32.1126	41.8370	32.8024	43.8126	33.2252	44.1050	33.3589	45.4443
	TransFT	36.0781	45.1084	37.8919	50.2874	38.6726	51.0883	38.7871	52.2503	39.4617	52.4747
	MetaST	31.6591	37.1050	32.4367	41.5279	32.9693	42.1080	33.0773	43.4723	33.1580	44.5011
	MetaTP-Lite	31.1476	37.3674	31.7718	40.5694	31.7011	42.0451	33.1686	43.8655	33.4906	45.3833
	MetaTP	**29.5126**	**35.1047**	**31.2094**	**40.0013**	**31.2905**	**40.6207**	**31.9758**	**41.3637**	**32.3755**	**43.3635**
	Methods	15 min		30 min		45 min		60 min		75 min	
		MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
Prediction length	HA	33.6614	45.9012	34.3882	46.4842	34.7708	47.3231	34.9184	49.9459	35.8997	51.9505
	VAR	32.9690	43.2544	33.4326	43.9375	34.5750	47.1259	34.8333	47.7916	35.2945	48.5648
	LR	45.7119	63.5798	48.7300	66.8309	51.0449	67.9682	53.0381	68.8936	55.7797	72.1186
	GP	34.4776	45.9649	35.3581	47.4044	35.2073	48.0301	36.0585	49.8693	36.0489	51.0505
	MAML-M	32.0624	41.6025	32.8542	43.8750	33.1003	44.4224	33.2814	44.6440	32.2793	44.8432
	MAML-F	31.2020	40.4384	31.9142	42.4919	32.1179	43.1012	32.5263	43.3987	33.0371	44.5992
	TransFT	34.9432	47.4424	37.0193	49.4063	38.2631	49.5776	38.5452	50.7517	39.0538	51.8123
	MetaST	31.1465	39.5452	31.6510	39.7491	31.5582	41.6709	32.3768	42.2470	32.9341	42.7864
	MetaTP-Lite	29.7784	37.9748	31.2016	39.5857	31.6315	41.4493	32.0123	41.7320	32.1685	42.1605
	MetaTP	**29.1061**	**35.6843**	**30.6595**	**38.9224**	**30.7202**	**39.1585**	**30.8582**	**40.8466**	**31.5213**	**41.3707**

One possible reason is that when p is too high, it becomes harder for VAR to capture the accurate periodical information from the source data.

TransFT doesn't perform very well. Meta-learning based approaches, i.e., MAML-M, MAML-F, and MetaST, achieve better performances than TransFT since they are able to learn better generalizable parameters, and adapt effectively to target regions. Among them, MetaST usually achieves better performance than MAML-F, which illustrates the effectiveness of the memory module of MetaST. But because MetaST lacks the interpolation network and the temporal memory, it is outperformed by our proposed method. MAML-F generally performs better than MAML-M. This indicates that the forward filling could be a better choice than filling with mean value, since the traffic data at each timestamp tend to be closer to the value of previous timestamp than to the mean.

5.2.3 Performance w.r.t. Prediction Length $\Delta t'$. In real-world applications, we often want to know the prediction results for further ahead in addition to the next time point. Therefore, in this section, we run all models for tasks with $\Delta t'$ from 1 to 5 (i.e., from 15 to 75 minutes). Except for the prediction length, other conditions are kept the same, i.e., Δt is 8 and p is fixed to be 60%. The results have been given in Table 1. From the table, we can see that for each individual method, the performance worsens (i.e., RMSE increases) when the prediction length gets longer. This is due to the fact that the traffic of nearer time points could be more related to the current time, while traffic for further timestamps tends to be impacted by more factors and get less related to current time. In other words, the further the time points are, the more uncertainties there are, and usually, the harder it is to predict. As a result, the prediction performances drop as $\Delta t'$ increases. MetaTP yields the best results under all choices of prediction lengths.

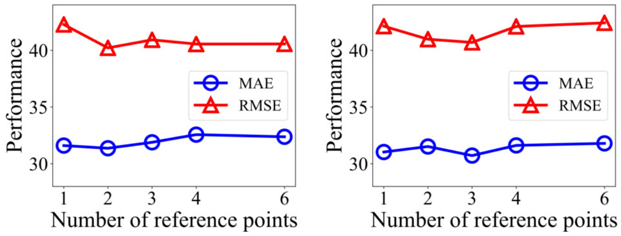


(a) Δt=10, $\Delta t'$=3, p=30% (b) Δt=10, $\Delta t'$=3, p=60% (c) Δt=5, $\Delta t'$=3, p=60%

Fig. 7. Performance w.r.t. the number of reference points on Stampede dataset.

5.2.4 Number of Reference Points K. Interpolation plays an important role in solving the data sparsity issue, and its effectiveness may be impacted by the number of reference points, i.e., K, that the input data is transformed onto. Thus, here we study the model performance with respect to the number of reference points K. A good choice of K may not only be related to Δt, but also have something to do with p. Therefore, we show the relationships between performance and K under three different settings in Figure 7a, Figure 7b and Figure 7c. Δt, $\Delta t'$ and p for Figure 7a are 10, 3 and 30%, respectively. And these three parameters for Figure 7b and Figure 7c are 10, 3, 60% and 5, 3, 60%, respectively. From these figures, we can see that under all 3 settings, the optimal choices of K are around Δt, and the performance gets worse when K gets too small or too large. The possible reason is that too few reference points may not be enough to hold all important information of the input nor the cross time series relationships. On the other hand, if there are too many reference points, then during the interpolation process, useless information, and even harmful noises could be introduced so that to generate a bad interpolated result which eventually decreases the model performance.

One might notice that the "best" performances from these figures are not exactly the same as those in Table 1. The reason is that during the calculation of these figures, to compete fairly, all settings are kept the same except for K. Therefore, results in these figures may not be the global optimal for the same setting of data sequence length, prediction length and missing rate. While in making the table, for each setting, we try to find its optimal result by trying out different combinations of other parameters.



(a) Δt=8, $\Delta t'$=2, p=60% (b) Δt=8, $\Delta t'$=3, p=60%

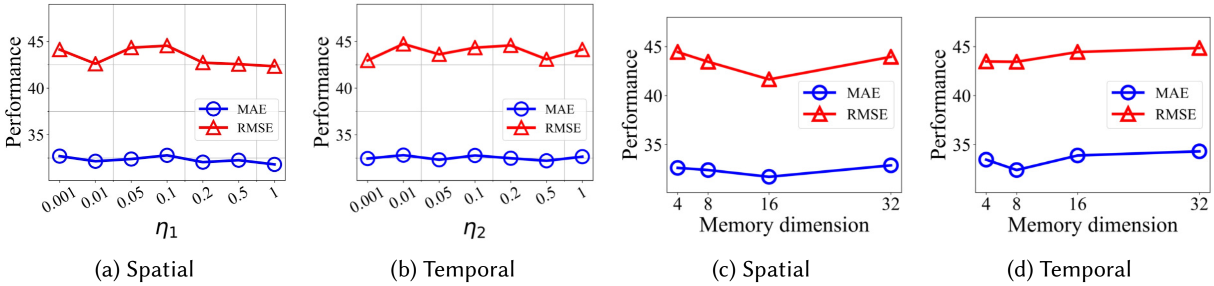
Fig. 8. Performance w.r.t. the number of reference points of intermediate prediction on Stampede dataset.

5.2.5 Number of Reference Points of Intermediate Prediction K′. As we mentioned in Section 4.1.2, we obtain the intermediate prediction from hidden states of the LSTM layer, before we use the second interpolation module to map data back onto the same time points as the prediction sequence. Thus, the number of reference points of the intermediate prediction, i.e., K', may impact the final performance. Therefore, we study the performances of different K's under different prediction lengths, and depict the results in Figure 8. Beside of the prediction length, other settings are the same, i.e., Δt is 8 and p is 60%. From these two figures, we can see that the best performances are both achieved when K' is the same as $\Delta t'$. When the K' decreases or increases, the performance gets worse. Similar to the study of K in Section 5.2.4, the choice of K' may also be impacted by the choices of Δt and p which we leave to future work.

5.2.6 *Memory Networks.*

Memory networks are used in this paper to capture global spatial and temporal correlations. We have already shown in Section 5.2.2 and Section 5.2.3 that the memory modules indeed have a positive impact on the overall performance. In this section, we study the memory networks from two aspects, i.e., impacts of loss weights (i.e., η_1 and η_2) and the memory size (i.e., C and C').

Loss Weights of Memory Modules. Figure 9a shows the changes of MAE and RMSE with different η_1, whereas Figure 9b is for η_2. When studying η_1, η_2 is fixed to be 0.1, and vice versa. From the figures we can see that the performances fluctuate in a relatively small range for different loss weights which indicates that the model is not very sensitive to them.



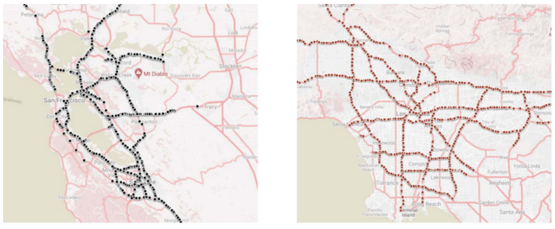
(a) Spatial (b) Temporal (c) Spatial (d) Temporal

Fig. 9. Performance w.r.t. loss weights and dimensions of memory networks on Stampede dataset.

Memory Size. Usually the dimension of memory could affect the effectiveness of memory modules. Therefore, in this section, we study the pattern that the performance may follow when the dimension of memory modules varies. Figure 9c and Figure 9d show changes of performance for different spatial and temporal memory dimensions, respectively. For both spatial and temporal, performances get worse when the dimension is too small because a too small memory network cannot hold enough information to provide help for queries. The same trend also holds for too large dimension which might because the memory has trouble picking up the more useful information to store, and trying saving too much information causes overfitting. The optimal dimensions for spatial and temporal are different which might because the temporal relationship is easier to capture than the spatial since it is harder to cluster nodes due to the difficulty in extracting useful features for clustering.

5.3 Traffic Speed Prediction on PeMS Dataset

In this section, we carry out experiments on PeMS dataset [7], which has been widely used in traffic prediction works [20, 28, 55]. These traffic data of California highway are collected by the Caltrans Performance Measurement System (PeMS) in real time every 30 seconds, and are aggregated into several intervals, e.g., 5 minutes and 30 minutes. There are two reasons that we use PeMS dataset. First, in previous section, we study the performance of MetaTP on local road network. But the traffic pattern of freeway can be different from local roads. Therefore, by testing with PeMS data, we can demonstrate the effectiveness of MetaTP working in different environments. Second, PeMS dataset does not have missingness, so that the prediction performance can be evaluated against more ground truths. We collect PeMS traffic speed data of 5 minutes interval of district 04, 06, 07, 08, and 11, from January 1, 2020 to January 31, 2020. The locations of stations from district 04 and 07 are depicted in Figure 10. Data from district 11 are used for testing, and others are used for training.



District 04

District 07

Fig. 10. Locations of stations from PeMS dataset.

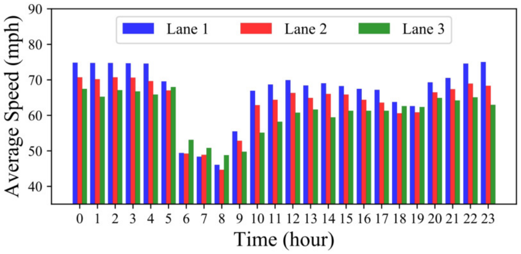


Fig. 11. Traffic speed of one station from PeMS dataset.

Figure 11 plots the average speed of three lanes from station 716088 of district 07. Since there are too many intervals to display in a single figure, for better visualization, here we only show values from the first interval of every hour. On the highway, lane 1, 2 and 3 are numbered from left to right (when facing towards the traffic moving direction). From the figure, we can observe the relationship among different lanes and the overall speed change along the time. The interval level average speed of lane 1 is the fastest, whereas lane 3 is usually the slowest. Morning rush hour and evening rush hour can be identified clearly from the figure. And the speed change between adjacent intervals is often very smooth (since we only plot the first interval of each hour, the change between adjacent intervals from the figure is larger than the real change between two consecutive intervals). In Section 5.3, unless otherwise stated, the default values of Δt, $\Delta t'$ and p are 24, 3, and 60%, respectively.

5.3.1 Performance Evaluation. Here we conduct experiments with different p and $\Delta t'$ on PeMS dataset. Results are listed in Table 2.

Missing Rate p. We can see that MetaTP outperforms all baselines. Compared with Stampede dataset, TransFT cannot perform well. The reason might be that the input sequence is longer here, and spatial-temporal patterns between source regions and target regions are quite different, making it hard to effectively transfer useful knowledge. As for LR, it achieve relative better performance. The reasons might be that the longer input sequence helps it capture the trend of a time series better, and also the traffic condition on highway changes more smoothly when compared with that on the local road network.

Prediction Length $\Delta t'$. From Table 2, we can see that MetaTP achieves the best performances for all different $\Delta t'$. When $\Delta t'$ gets larger, the performances of all methods decrease, and the performance gaps between MetaTP and baselines generally increase. The reason might be that the prediction lengths used here are longer, which brings more challenging. The global spatial and temporal relationships that are captured by the proposed memory networks help to improve the performance.

5.3.2 Study of Number of Reference Points. The same with Stampede dataset, here, we study the performances for different choices of number of reference points, including K and K'.

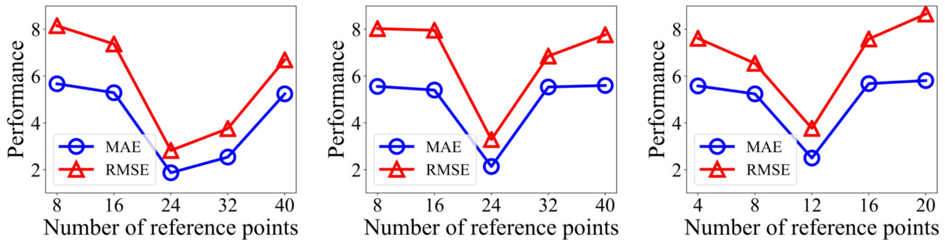
Number of Reference Points K. Figure 12 shows changes of performances with respect to K under different settings. It can be found from these figures that the best performances are usually achieved when K is set to be around Δt.

Number of Reference Points of Intermediate Prediction K'. The results are shown in Figure 13. From the figures we can see that under both conditions, the best performances are achieved when K' is around $\Delta t'$.

5.3.3 Memory. In this section, we study the memory networks from following aspects, i.e., loss weights (i.e., η_1 and η_2), memory size and the visualization of attention scores.

Table 2. Performances on PeMS dataset w.r.t. missing rate and prediction length.

	Methods	10%		30%		50%		70%		90%	
		MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
Missing rate	HA	2.4320	5.7797	2.5244	6.0528	2.5266	6.0950	2.5864	6.2448	2.6292	6.3822
	VAR	2.2450	5.2744	2.3758	5.6922	2.4723	5.8684	2.5371	6.1239	2.5887	6.2795
	LR	2.0682	4.5854	2.1429	4.8816	2.2052	5.1181	2.3590	5.5698	2.5378	6.0892
	GP	2.0368	4.3313	2.0864	4.4502	2.1773	4.7764	2.2682	5.0214	2.4769	5.7732
	MAML-M	3.5993	5.4107	3.7242	5.5487	3.9235	5.8092	4.0172	5.8650	4.0172	6.1695
	MAML-F	3.5304	5.3156	3.7134	5.3817	3.7662	5.4796	3.8500	5.7177	3.9918	5.9263
	TransFT	5.2691	10.5968	5.3002	10.6504	5.3104	10.6551	5.4074	10.8821	5.8046	11.1510
	MetaST	2.5531	3.7252	2.6252	4.0849	2.8410	4.5373	2.9125	4.6630	2.9419	4.8231
	MetaTP-Lite	1.6901	2.7388	1.7997	2.8095	1.8685	2.9627	1.9358	3.0009	2.2529	3.6267
	MetaTP	**1.5598**	**2.2708**	**1.7019**	**2.4066**	**1.7250**	**2.6042**	**1.8760**	**2.8196**	**1.9768**	**3.1845**
	Methods	20 min		30 min		40 min		50 min		60 min	
		MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
Prediction length	HA	2.5430	6.0576	2.6553	6.3217	2.7747	6.5897	2.9349	7.0460	2.9649	7.0893
	VAR	2.4654	5.8718	2.5006	5.9150	2.5401	6.0179	2.8235	6.5376	2.9071	6.8914
	LR	2.2818	5.1900	2.5313	5.7968	2.8043	6.5529	2.9353	6.7161	3.1390	7.4470
	GP	2.3157	5.1192	2.5479	5.5674	2.7390	5.9644	2.9347	6.4975	3.0817	6.7906
	MAML-M	3.7182	5.7448	3.6928	5.8580	3.7881	5.9852	4.2919	6.3766	4.4497	6.6952
	MAML-F	3.7067	5.7574	3.7526	5.8509	3.9975	5.9717	4.0131	6.2837	4.1382	6.5153
	TransFT	5.3191	10.6516	5.4996	10.8005	5.5090	11.0346	5.9529	11.2635	6.0363	12.4115
	MetaST	2.8434	4.4496	3.0947	4.4871	3.1020	5.0799	3.2218	5.4403	3.8200	6.1732
	MetaTP-Lite	1.8606	2.8379	2.0235	3.0443	2.4293	3.7075	2.5954	3.8050	2.6444	4.2053
	MetaTP	**1.6067**	**2.6045**	**1.9199**	**2.7277**	**1.9359**	**2.9075**	**2.2798**	**3.3700**	**2.3429**	**3.6073**

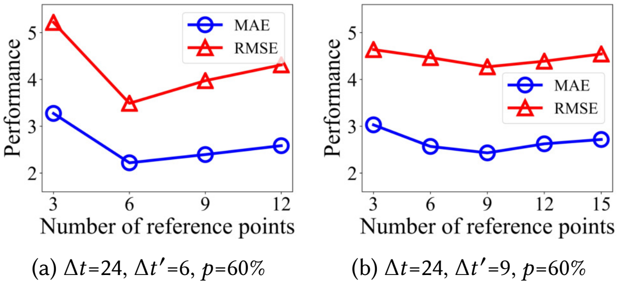


(a) Δt=24, $\Delta t'$=3, p=20%　(b) Δt=24, $\Delta t'$=3, p=60%　(c) Δt=12, $\Delta t'$=3, p=60%

Fig. 12. Performance w.r.t. the number of reference points on PeMS dataset.

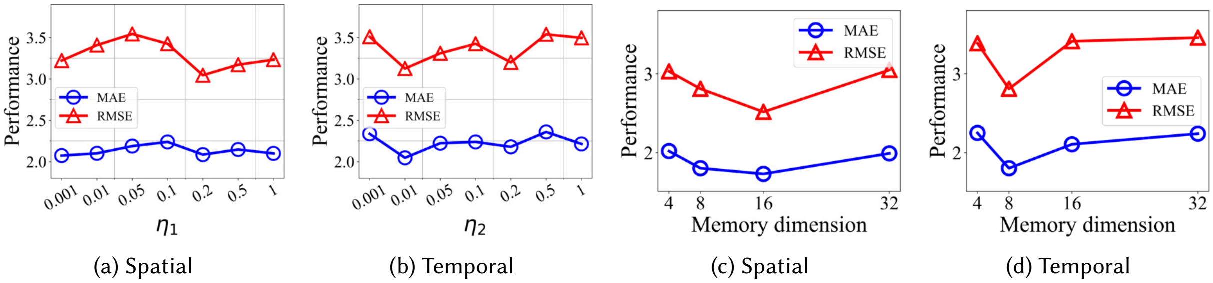
Loss Ratio of Memory Modules. Figure 14a and 14b show performances of MetaTP on PeMS dataset with different choices of η_1 and η_2, respectively. The best performance for η_1 is achieved at around 0.2, whereas for η_2, it is 0.01.

Memory Dimension. Figure 14c and 14d show the performances with different spatial and temporal memory dimensions, respectively. Similar to Stampede, the best performances are achieved when the spatial dimension is 16 and the temporal dimension is 8.



(a) Δt=24, $\Delta t'$=6, p=60% (b) Δt=24, $\Delta t'$=9, p=60%

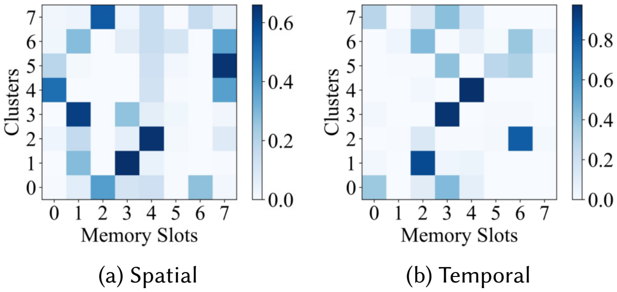
Fig. 13. Performance w.r.t. the number of reference points of intermediate prediction on PeMS dataset.



(a) Spatial (b) Temporal (c) Spatial (d) Temporal

Fig. 14. Performance w.r.t. loss ratios and dimensions of memory networks on PeMS dataset.

Visualization of Memory. In order to investigate the knowledge patterns captured by memory networks, we randomly select data examples from each spatial cluster and calculate their attention scores based on Eq. (13). The results for spatial and temporal memories are shown in Figure 15a and Figure 15b, respectively. The y-axis is the id of cluster and the x-axis is the memory slot number. The color in the images indicates the relationships between each example and each memory slot. The darker the color, the stronger the relationship. From figures we can see that examples from different clusters tend to have different attention weights across memory slots. This indicates that memories have captured certain patterns for different spatial and temporal clusters, and they help to get a better and enhanced data representation.



(a) Spatial (b) Temporal

Fig. 15. Visualization of attention scores on PeMS dataset.

6 DISCUSSION

In this section, we give discussions on three aspects. We first discuss the lessons learned from the data collection process using our deployed system. Then we talk about some of the remaining challenges in the studied problem and the potential future work. In the end, we detail our contributions to the Ubicomp community.

6.1 Lesson Learned

Real-world data acquisition system needs to be sufficiently robust in order for large scale deployment. In our case, we wish to eliminate the need of human intervention as much as possible after the GPS tracking device has been installed in the Stampede shuttle. Since these shuttles are operated by a third party company, and the smartphones are placed in the locked overhead bin, it is impractical to frequently visit the third party company to check the running status of each device. There might be various reasons for the failure of data upload from devices, e.g., the shuttle is out of service, the application crashes, the phone has a dead battery due to freezing weather or overheating. In order to make the system more robust, we root the smartphone so that whenever it connects to power (indicating the shuttle engine starts), the data collection application launches automatically. And if the phone is in power-off mode, the connection to power will trigger the boot-up of the smartphone, and then launch the application.

In addition to GPS data, we also collect other sensory data using the same application, including data from accelerometer and gyroscope. In this case, we can obtain accurate running information of the shuttles by making use of the other sensory data along with the priori knowledge about Stampede routes, even if the GPS signals were impacted by urban canyon effect. During our data collection process, the GPS signal was in overall good quality since there are no tall buildings in the operating area of Stampede shuttles. So during the experiments, we can extract the accurate travel time on each road segments from the collected data without using data from other sensors.

6.2 Future Work

Our current model predicts traffic conditions for each target road segment only using the historical data from the same road segment. This is based on the consideration that the spatial correlations are less helpful for sparse and irregular input time series data since many neighbors may have very few amount of collected data. However, we can use our interpolation algorithm to generate complete time series data for all the road segments, which enables the use of more advanced spatial-temporal predictive models, such as graph neural network [15, 31, 55, 58, 59], to further improve the prediction accuracy. What's more, we can infer the traffic conditions for road segments that have no historical data, by using the data from their neighbors and utilizing their relationships with neighbors, e.g., traffic cascading [14, 29, 30]. This iteration process can keep going until all road segments in the target regions are covered. Depending on prediction results, we can choose to deploy more roving sensors to locations whose performances cannot keep up with others, and deploy fewer sensors to areas that are less impacted by a lower frequency of data collection.

Another aspect we can try is to use mixed sources of data. Currently, due to the different characteristics of local roads and freeway, we use the data from local road networks to predict local traffic, and use freeway data to make prediction for freeway. Since there are a large amount of collected data from freeway, and certain traffic patterns could be shared between local and freeway, we can try to use the freeway data to help improve the prediction performance of local roads.

6.3 Contributions to the Community

Solving the traffic prediction task could help on building the intelligent transportation system. In particular, it could improve the delivery efficiency and the accuracy of delivery time estimation so as to improve customer satisfaction with shipping services, including online grocery shopping, e.g., Instacart, and food delivery service, e.g., DoorDash and UberEats. Since it is not always possible for each company to collect enough data for each target location to train a separate model directly and effectively, our proposed model can be used to effectively make traffic prediction by using only a small amount of data samples.

In addition to traffic prediction, the proposed model can be easily adopted to solve prediction problems in other domains involving spatio-temporal data, e.g., water quality forecasting [53], forecasting for gas price, weather, and air quality [34], and even forecasting for the arriving time of flights [26]. Because the spatial and temporal sparsity challenge may exist in these applications as well.

We will make the collected Stampede dataset public to facilitate the research in this area. The whole dataset includes GPS data from shuttles running in the local road network with a high sampling rate for a duration of over two years. We hope that the dataset would help researchers to gain more insights of fine-grained traffic patterns of local road networks.

7 CONCLUSION

In this paper, we propose a meta-learning based framework to perform traffic prediction on individual road segments. The nature of roving sensor brings new challenges: collected data is temporally sparse and unevenly distributed across regions. To overcome the challenges, we employ an interpolation network to handle irregularly-spaced time series and enable temporal correlations to be effectively captured; we employ meta-learning to transfer knowledge from source to target regions with limited available data via fast adaptation. Spatial and temporal memory networks are developed to further capture global patterns among regions and timestamps. This framework can also be applied to other related problems, such as air quality and weather forecasting, to overcome the issues of irregularly distributed time series and limited available data.

ACKNOWLEDGMENTS

This work was supported in part by the US National Science Foundation under Grants CNS-1652503 and CNS-1737590. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of NSF.

REFERENCES

[1] JM Angulo, W Gonzalez-Manteiga, M Febrero-Bande, and FJ Alonso. 1998. Semi-parametric statistical approaches for space-time process prediction. *Environmental and Ecological Statistics* 5, 4 (1998), 297–316.

[2] Matteo Biasetton, Umberto Michieli, Gianluca Agresti, and Pietro Zanuttigh. 2019. Unsupervised domain adaptation for semantic segmentation of urban scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*.

[3] Wei Cao, Dong Wang, Jian Li, Hao Zhou, Lei Li, and Yitan Li. 2018. Brits: Bidirectional recurrent imputation for time series. *arXiv preprint arXiv:1805.10572* (2018).

[4] Di Chai, Leye Wang, and Qiang Yang. 2018. Bike flow prediction with multi-graph convolutional networks. In *Proceedings of the 26th ACM SIGSPATIAL international conference on advances in geographic information systems*. 397–400.

[5] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. 2018. Recurrent neural networks for multivariate time series with missing values. *Scientific reports* 8, 1 (2018), 1–12.

[6] Cen Chen, Kenli Li, Sin G Teo, Xiaofeng Zou, Kang Wang, Jie Wang, and Zeng Zeng. 2019. Gated residual recurrent graph neural networks for traffic prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 485–492.

[7] Chao Chen, Karl Petty, Alexander Skabardonis, Pravin Varaiya, and Zhanfeng Jia. 2001. Freeway performance measurement system: mining loop detector data. *Transportation Research Record* 1748, 1 (2001), 96–102.

[8] Jie Chen, Kian Hsiang Low, Yujian Yao, and Patrick Jaillet. 2015. Gaussian process decentralized data fusion and active sensing for spatiotemporal traffic modeling and prediction in mobility-on-demand systems. *IEEE Transactions on Automation Science and Engineering* 12, 3 (2015), 901–921.

[9] Yun Cheng, Xiaoxi He, Zimu Zhou, and Lothar Thiele. 2019. Ict: In-field calibration transfer for air quality sensor deployments. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3, 1 (2019), 1–19.

[10] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[11] Younggeun Cho and Jungsuk Kwac. 2007. A Travel Time Prediciton with Machine Learning Algorithms. *Machine Learning-Project (CS 229)* (2007).

[12] Stephen Clark. 2003. Traffic prediction using multivariate nonparametric regression. *Journal of transportation engineering* 129, 2 (2003), 161–168.

[13] Zhiyong Cui, Kristian Henrickson, Ruimin Ke, and Yinhai Wang. 2019. Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting. *IEEE Transactions on Intelligent Transportation Systems* (2019).

[14] Xiaolei Di, Yu Xiao, Chao Zhu, Yang Deng, Qinpei Zhao, and Weixiong Rao. 2019. Traffic congestion prediction by spatiotemporal propagation patterns. In *2019 20th IEEE International Conference on Mobile Data Management (MDM)*. IEEE, 298–303.

[15] Zulong Diao, Xin Wang, Dafang Zhang, Yingru Liu, Kun Xie, and Shaoyao He. 2019. Dynamic spatial-temporal graph convolutional neural networks for traffic forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 890–897.

[16] Loan NN Do, Hai L Vu, Bao Q Vo, Zhiyuan Liu, and Dinh Phung. 2019. An effective spatial-temporal attention based neural network for traffic flow prediction. *Transportation research part C: emerging technologies* 108 (2019), 12–28.

[17] Zhihan Fang, Yu Yang, Shuai Wang, Boyang Fu, Zixing Song, Fan Zhang, and Desheng Zhang. 2019. MAC: Measuring the impacts of anomalies on travel time of multiple transportation systems. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3, 2 (2019), 1–24.

[18] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*. 1126–1135.

[19] Taesik Gong, Yeonsu Kim, Jinwoo Shin, and Sung-Ju Lee. 2019. MetaSense: few-shot adaptation to untrained conditions in deep mobile sensing. In *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*. 110–123.

[20] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. 2019. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 922–929.

[21] Yunhui Guo, Honghui Shi, Abhishek Kumar, Kristen Grauman, Tajana Rosing, and Rogerio Feris. 2019. Spottune: transfer learning through adaptive fine-tuning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4805–4814.

[22] Tianfu He, Jie Bao, Ruiyuan Li, Sijie Ruan, Yanhua Li, Li Song, Hui He, and Yu Zheng. 2020. What is the Human Mobility in a New City: Transfer Mobility Knowledge Across Cities. In *Proceedings of The Web Conference 2020*. 1355–1365.

[23] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[24] Weiwei Jiang and Jiayun Luo. 2021. Graph neural network for traffic forecasting: A survey. *arXiv preprint arXiv:2101.11174* (2021).

[25] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[26] Qi Li, Yaliang Li, Jing Gao, Bo Zhao, Wei Fan, and Jiawei Han. 2014. Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 1187–1198.

[27] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *International Conference on Learning Representations (ICLR '18)*.

[28] Zhi Li, Yanmin Zhu, Hongzi Zhu, and Minglu Li. 2011. Compressive sensing approach to urban traffic sensing. In *2011 31st international conference on distributed computing systems*. IEEE, 889–898.

[29] Yuxuan Liang, Zhongyuan Jiang, and Yu Zheng. 2017. Inferring traffic cascading patterns. In *Proceedings of the 25th acm sigspatial international conference on advances in geographic information systems*. 1–10.

[30] Yuxuan Liang, Songyu Ke, Junbo Zhang, Xiuwen Yi, and Yu Zheng. 2018. Geoman: Multi-level attention networks for geo-sensory time series prediction.. In *IJCAI*. 3428–3434.

[31] Wenjuan Luo, Han Zhang, Xiaodi Yang, Lin Bo, Xiaoqing Yang, Zang Li, Xiaohu Qie, and Jieping Ye. 2020. Dynamic Heterogeneous Graph Neural Network for Real-time Event Prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 3213–3223.

[32] Yonghong Luo, Ying Zhang, Xiangrui Cai, and Xiaojie Yuan. 2019. E2gan: End-to-end generative adversarial network for multivariate time series imputation. In *AAAI Press*. 3094–3100.

[33] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. 2014. Traffic flow prediction with big data: a deep learning approach. *IEEE Transactions on Intelligent Transportation Systems* 16, 2 (2014), 865–873.

[34] Chuishi Meng, Wenjun Jiang, Yaliang Li, Jing Gao, Lu Su, Hu Ding, and Yun Cheng. 2015. Truth discovery on crowd sensing of correlated entities. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. 169–182.

[35] Wanli Min and Laura Wynter. 2011. Real-time road traffic prediction with spatio-temporal correlations. *Transportation Research Part C: Emerging Technologies* 19, 4 (2011), 606–616.

[36] Zheyi Pan, Yuxuan Liang, Weifeng Wang, Yong Yu, Yu Zheng, and Junbo Zhang. 2019. Urban traffic prediction from spatio-temporal data using deep meta learning. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1720–1730.

[37] Zheyi Pan, Wentao Zhang, Yuxuan Liang, Weinan Zhang, Yong Yu, Junbo Zhang, and Yu Zheng. 2020. Spatio-Temporal Meta Learning for Urban Traffic Prediction. *IEEE Transactions on Knowledge and Data Engineering* (2020).

[38] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*. 8026–8037.

[39] Ling Shao, Fan Zhu, and Xuelong Li. 2014. Transfer learning for visual categorization: A survey. *IEEE transactions on neural networks and learning systems* 26, 5 (2014), 1019–1034.

[40] Satya Narayan Shukla and Benjamin Marlin. 2019. Interpolation-Prediction Networks for Irregularly Sampled Time Series. In *International Conference on Learning Representations*.

[41] Vanessa J Stauch and Andrew J Jarvis. 2006. A semi-parametric gap-filling model for eddy covariance CO2 flux time series data. *Global Change Biology* 12, 9 (2006), 1707–1716.

[42] Man-Chun Tan, Sze Chun Wong, Jian-Min Xu, Zhan-Rong Guan, and Peng Zhang. 2009. An aggregation approach to short-term traffic flow prediction. *IEEE Transactions on Intelligent Transportation Systems* 10, 1 (2009), 60–69.

[43] Xianfeng Tang, Huaxiu Yao, Yiwei Sun, Charu Aggarwal, Prasenjit Mitra, and Suhang Wang. 2020. Joint modeling of local and global temporal dynamics for multivariate time series forecasting with missing values. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 5956–5963.

[44] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. 2017. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7167–7176.

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[46] Jin Wang and Qixin Shi. 2013. Short-term traffic speed forecasting hybrid model based on chaos–wavelet analysis-support vector machine theory. *Transportation Research Part C: Emerging Technologies* 27 (2013), 219–232.

[47] Yibing Wang, Markos Papageorgiou, and Albert Messmer. 2005. A real-time freeway network traffic surveillance tool. *IEEE Transactions on control systems technology* 14, 1 (2005), 18–32.

[48] Yilun Wang, Yu Zheng, and Yexiang Xue. 2014. Travel time estimation of a path using sparse trajectories. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 25–34.

[49] Ying Wei, Yu Zheng, and Qiang Yang. 2016. Transfer knowledge between cities. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1905–1914.

[50] Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916* (2014).

[51] Xiaoyang Xie, Yu Yang, Zhihan Fang, Guang Wang, Fan Zhang, Fan Zhang, Yunhuai Liu, and Desheng Zhang. 2018. coSense: Collaborative urban-scale vehicle sensing based on heterogeneous fleets. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 4 (2018), 1–25.

[52] Yuanchang Xie, Kaiguang Zhao, Ying Sun, and Dawei Chen. 2010. Gaussian processes for short-term traffic volume forecasting. *Transportation Research Record* 2165, 1 (2010), 69–78.

[53] Huaxiu Yao, Yiding Liu, Ying Wei, Xianfeng Tang, and Zhenhui Li. 2019. Learning from multiple cities: A meta-learning approach for spatial-temporal prediction. In *The World Wide Web Conference*. 2181–2191.

[54] Huaxiu Yao, Xianfeng Tang, Hua Wei, Guanjie Zheng, and Zhenhui Li. 2019. Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 5668–5675.

[55] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2018. Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 3634–3640.

[56] Haiyang Yu, Zhihai Wu, Shuqin Wang, Yunpeng Wang, and Xiaolei Ma. 2017. Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks. *Sensors* 17, 7 (2017), 1501.

[57] Chuanting Zhang, Haixia Zhang, Jingping Qiao, Dongfeng Yuan, and Minggao Zhang. 2019. Deep transfer learning for intelligent cellular traffic prediction based on cross-domain big data. *IEEE Journal on Selected Areas in Communications* 37, 6 (2019), 1389–1401.

[58] Junbo Zhang, Yu Zheng, Dekang Qi, Ruiyuan Li, Xiuwen Yi, and Tianrui Li. 2018. Predicting citywide crowd flows using deep spatio-temporal residual networks. *Artificial Intelligence* 259 (2018), 147–166.

[59] Qi Zhang, Jianlong Chang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. 2020. Spatio-Temporal Graph Structure Learning for Traffic Forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 1177–1185.

[60] Yan Zhang, Yunhuai Liu, Genjian Li, Yi Ding, Ning Chen, Hao Zhang, Tian He, and Desheng Zhang. 2019. Route Prediction for Instant Delivery. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3, 3 (2019), 1–25.

[61] Yang Zhang, Hongxiao Wang, Daniel Zhang, and Dong Wang. 2019. Deeprisk: A deep transfer learning approach to migratable traffic risk estimation in intelligent transportation using social sensing. In *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE, 123–130.

[62] Jing Zhao and Shiliang Sun. 2016. High-order Gaussian process dynamical models for traffic flow prediction. *IEEE Transactions on Intelligent Transportation Systems* 17, 7 (2016), 2014–2019.

[63] Ling Zhao, Yujiao Song, Chao Zhang, Yu Liu, Pu Wang, Tao Lin, Min Deng, and Haifeng Li. 2019. T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE Transactions on Intelligent Transportation Systems* (2019).

[64] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. 2014. Urban computing: concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)* 5, 3 (2014), 1–55.

[65] Weida Zhong, Qiuling Suo, Xiaowei Jia, Aidong Zhang, and Lu Su. 2021. Heterogeneous Spatio-Temporal Graph ConvolutionNetwork for Traffic Forecasting with Missing Values. In *2021 IEEE 41th International Conference on Distributed Computing Systems (ICDCS)*. IEEE.

[66] Weida Zhong, Qiuling Suo, Fenglong Ma, Yunfei Hou, Abhishek Gupta, Chunming Qiao, and Lu Su. 2019. A reliability-aware vehicular crowdsensing system for pothole profiling. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3, 4 (2019), 1–26.