Urban Traffic Dynamics Prediction—A Continuous Spatial-temporal Meta-learning Approach

YINGXUE ZHANG and YANHUA LI, Worcester Polytechnic Institute, USA
XUN ZHOU, University of Iowa, USA
JUN LUO, Lenovo Group Limited, Hong Kong
ZHI-LI ZHANG, University of Minnesota-Twin Cities, USA

Urban traffic status (e.g., traffic speed and volume) is highly dynamic in nature, namely, varying across space and evolving over time. Thus, predicting such traffic dynamics is of great importance to urban development and transportation management. However, it is very challenging to solve this problem due to spatial-temporal dependencies and traffic uncertainties. In this article, we solve the traffic dynamics prediction problem from Bayesian meta-learning perspective and propose a novel continuous spatial-temporal meta-learner (cST-ML), which is trained on a distribution of traffic prediction tasks segmented by historical traffic data with the goal of learning a strategy that can be quickly adapted to related but unseen traffic prediction tasks. cST-ML tackles the traffic dynamics prediction challenges by advancing the Bayesian black-box meta-learning framework through the following new points: (1) cST-ML captures the dynamics of traffic prediction tasks using variational inference, and to better capture the temporal uncertainties within tasks, cST-ML performs as a rolling window within each task; (2) cST-ML has novel designs in architecture, where CNN and LSTM are embedded to capture the spatial-temporal dependencies between traffic status and traffic-related features; (3) novel training and testing algorithms for cST-ML are designed. We also conduct experiments on two real-world traffic datasets (taxi inflow and traffic speed) to evaluate our proposed cST-ML. The experimental results verify that cST-ML can significantly improve the urban traffic prediction performance and outperform all baseline models especially when obvious traffic dynamics and temporal uncertainties are presented.

CCS Concepts: • **Information systems → Spatial-temporal systems**; • **Computing methodologies → Neural networks;**

Additional Key Words and Phrases: Traffic dynamics prediction, Bayesian meta-learning, spatial-temporal data

Yingxue Zhang and Yanhua Li were supported in part by NSF Grants No. IIS-1942680 (CAREER), No. CNS-1952085, No. CMMI-1831140, and No. DGE-2021871. Zhi-Li Zhang was supported in part by NSF Grants No. CNS-1952085, No. CMMI-1831140, and No. CNS-1901103. Xun Zhou is funded partially by Safety Research using Simulation University Transportation Center (SAFER-SIM). SAFER-SIM is funded by a grant from the U.S. Department of Transportation's University Transportation Centers Program (No. 69A3551747131). However, the U.S. Government assumes no liability for the contents or use thereof.

Authors' addresses: Y. Zhang and Y. Li, Worcester Polytechnic Institute, 100 Institute Rd, Worcester, MA 01609; emails: {yzhang31, yli15}@wpi.edu; X. Zhou, University of Iowa, Iowa City, IA 52242; email: xun-zhou@uiowa.edu; J. Luo, Lenovo Group Limited, Quarry Bay, King's Rd, Hong Kong; email: jluo1@lenovo.com; Z.-L. Zhang, University of Minnesota-Twin Cities, Minneapolis, MN 55455; email: zhzhang@cs.umn.edu.

2157-6904/2022/01-ART23 $15.00
https://doi.org/10.1145/3474837

ACM Reference format:
Yingxue Zhang, Yanhua Li, Xun Zhou, Jun Luo, and Zhi-Li Zhang. 2022. Urban Traffic Dynamics Prediction—A Continuous Spatial-temporal Meta-learning Approach. *ACM Trans. Intell. Syst. Technol.* 13, 2, Article 23 (January 2022), 19 pages.
https://doi.org/10.1145/3474837

1 INTRODUCTION

Over the past a few decades, the rapid population growth has accelerated the process of urbanization, which in turn brings huge impacts on the urban traffic including the increasing traffic volume, worse traffic condition and the overload of the transportation infrastructures. As a result, accurately predicting the *highly dynamic traffic status* (e.g., traffic volume, speed, and inflow) has become a crucial work for urban development aiming to reduce congestion and increase mobility, since it can not only provide insights for urban planning, helping to improve the efficiency of public transportation, but also guarantee the public safety [30, 33].

Given the underlying road network and the historical traffic observations, *the problem of traffic dynamics prediction* aims at forecasting short-term traffic status in consecutive time slots. However, there are many practical challenges before solving this problem:
(1) *Spatial-temporal dependencies.* It is the most common challenge when dealing with traffic dynamics prediction problem, since the traffic status would be influenced by the nearby environments, road networks and its previous traffic status.
(2) *Traffic dynamics and temporal uncertainties.* In traffic dynamics prediction, the most difficult part is to capture and model the dynamics of traffic status, since urban traffic always contains temporal uncertainties due to sudden travel demand changes, unexpected events or extreme weather. For example, Figure 1 is an illustration of traffic dynamics, where it is possible that the traffic patterns are almost consistent in the first two days but show obvious fluctuations and temporal uncertainties in the next few days. The reasons of such considerable changes in traffic patterns could be a thunder storm, a large sports event or a car crash. In general, irregular and drastic traffic changes caused by these factors are hard to capture using traditional time series models due to their non-periodicity and rareness (i.e., lacking training samples).

A lot of research efforts have been put into the traffic dynamics prediction area. Some works [5, 14] use traditional machine learning methods and time series models to predict urban traffic. In addition, deep neural networks are also widely used in urban traffic prediction works such as [4, 8, 15, 29, 30]. However, these studies do not consider the situation where the traffic shows strong non-stationarity. Moreover, a few works are inspired by meta-learning and try to apply existing meta-learning methods to solve the traffic dynamic prediction problem, such as References [20, 24]. However, these works still do not consider the temporal uncertainties and have limited capabilities to learn traffic patterns that are rarely seen in the historical data.

In this article, we try to solve the short-term traffic dynamics prediction problem and tackle the unique challenges mentioned before from the Bayesian meta-learning perspective. We propose to predict the traffic using some traffic-related features (e.g., travel demands), since these features can provide more information about the spatial-temporal dependencies of traffic, and sometimes the changes of features also indicate the traffic dynamics. Thus, using traffic-related features potentially improves the accuracy of traffic prediction. Besides, a novel **continuous spatial-temporal meta-learner (cST-ML)**[1] is proposed, which is trained on a distribution of traffic prediction tasks generated by traffic time series data with the goal of learning a strategy

[1] A preliminary version of the results in this article appeared in Reference [31].

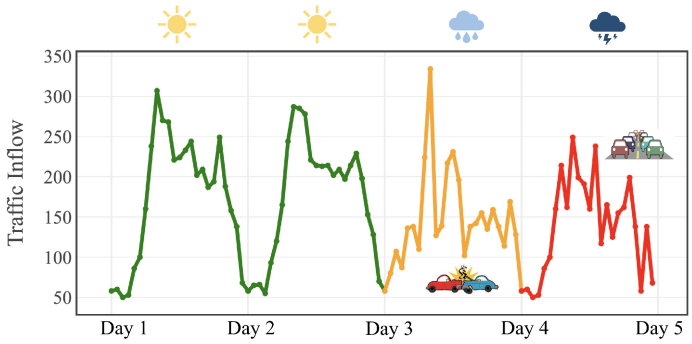


Fig. 1. Illustration of traffic dynamics.

that can be quickly generalized to related but unseen traffic prediction tasks from the same task distribution. cST-ML captures the spatial-temporal dependencies of traffic as well as the temporal uncertainties and dynamics through variational inference and rolling windows. Our **main contributions** are summarized as follows:

- We are the first to solve the traffic dynamics prediction problem from the Bayesian meta-learning perspective and propose a novel continuous spatial-temporal meta-learner cST-ML. cST-ML advances the Bayesian black-box meta-learning framework to capture traffic dynamics and temporal uncertainties (see Section 3.2).
- cST-ML features some novel designs in the architecture. cST-ML is composed of an inference network and a decoder, where CNN and LSTM are embedded to realize the goal of capturing traffic spatial-temporal dependencies. Novel algorithms are also designed for cST-ML training and testing. During meta-training and testing, in each task, cST-ML performs traffic prediction as a rolling window, which not only keeps the task uncertainties but also maintains the temporal uncertainties within each task (see Sections 3.3 and 3.4).
- We conduct extensive experiments on two real-world traffic datasets (taxi inflow and traffic speed) to evaluate our proposed cST-ML. The experimental results verify that cST-ML can significantly improve the urban traffic prediction performance and outperform all existing baseline methods on both datasets especially when obvious traffic dynamics and temporal uncertainties are presented (see Section 4). ***We have made our codes available to contribute to the research community [1].***
- Compared with the preliminary version of this work in Reference [31], (i) we analyze the state-of-the-art meta-learning methods and the key challenges of urban dynamics prediction problem (see Section 3.1). (ii) We improve the meta-learner model by adding a memory vector and a rolling window, which help to capture the inner temporal uncertainties within tasks and improve the prediction accuracy. In addition, the objective function, training and testing algorithms are also improved (see Sections 3.2, 3.3, and 3.4). (iii) We provide more evaluation results including 6-h prediction performance and the evaluations on hyper-parameters (see Section 4.5). (iv) To further illustrate the effectiveness of cST-ML, we add a case study and look into real traffic dynamics prediction cases (see Section 4.6). (v) We add comprehensive related work section to discuss and distinguish our work from the state-of-the-art studies (see Section 5).

The rest of the article is organized as follows: Section 2 provides the overview of the article, Section 3 details the design of cST-ML. We present the experimental results in Section 4 and discuss related work in Section 5. Finally, we conclude the article in Section 6.

Table 1. Notations

Notations	Descriptions
$\mathcal{S} = \{s_{ij}\}$	Grid cells
R_{ij}	A target region
$N_t \in \mathbb{N}$	Number of time slots in each task
$\tau \in \mathbb{N}$	Number of tasks
$\boldsymbol{X}^t$	Traffic-related features at t
y^t	Traffic status at t
w	Rolling window size
θ	Parameters of meta-learner
$\mathcal{T}_i = \{\mathcal{D}_i^{tr}, \mathcal{D}_i^{ts}\}$	One traffic prediction task
ϕ_i	Adapted parameters for task $\mathcal{T}_i$

2 OVERVIEW

In this section, we define the traffic dynamics prediction problem, and outline our solution framework. Table 1 lists the notations used in the article.

2.1 Problem Definition

In a city, *urban traffic status* can be characterized by many statistics, such as traffic volumes, speed, inflow/outflow, and so on, which are of great interest to urban planners and researchers for transportation planning, traffic evaluation, and more. These statistics are dynamic in nature, namely, varying across space and evolving over time. Hence, we divide an urban area into grid cells as defined below. Each grid cell represents a target area for *urban dynamics prediction*.

Definition 1 (Grid Cells). We divide a city into $I \times J$ grid cells with equal side-length (e.g., 1×1km), denoted as $\mathcal{S} = \{s_{ij}\}$, where $1 \le i \le I, 1 \le j \le J$.

Definition 2 (Target Region for a Target Grid Cell). For a target grid cell s_{ij}, its target region is a square geographic region with s_{ij} in center, formed by $\ell \times \ell$ grid cells, denoted with $R_{ij} = \langle s_{ij}, \ell \rangle$.

In our study, we assume the traffic status in a target grid cell s_{ij} has high spatial correlations with the other grid cells within its target region.

Definition 3 (Traffic-related Features). All features that will influence the traffic status are traffic-related features, e.g., time of the day, travel demand, and so on. For a grid cell s, we denote x^t as one feature of s in time slot t. For a target region R, we denote X^t as one feature map of R in time slot t, where X^t is a $\ell \times \ell$ matrix. Since there could be multiple features, all the feature maps in region R in time slot t can be denoted with a tensor $\boldsymbol{X}^t = \{X_1^t, \ldots, X_n^t\} \in \mathbb{R}^{n \times \ell \times \ell}$, where $n \in \mathbb{N}^+$ is the number of features.

Most of the traffic-related features are easy to acquire and can be extracted from multiple sources. Taking travel demands as an example, since it is hard to obtain the travel demands of all transportation modes, we can use taxi demands or bus demands instead, and many studies have shown that taxi and bus demand are representative measures of travel demand [10, 18].

Definition 4 (Traffic Status). Traffic status indicates the quality of traffic, which can be measured by traffic inflow/outflow, average driving speed, and so on. We denote y^t as the average traffic status of grid cell s in time slot t.

In this article, we choose one specific measure of traffic status as the target of prediction, other measures if available can be treated as traffic-related features during prediction.

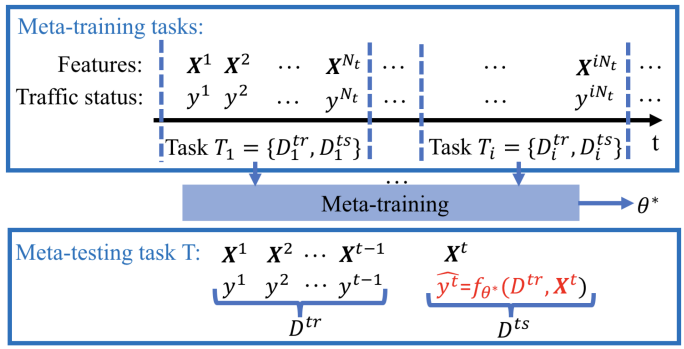


Fig. 2. Problem illustration.

Definition 5 (Traffic Prediction Task). A traffic prediction task $\mathcal{T}_i$ is composed of a set of paired (X^t, y^t) in N_t consecutive time slots, which is divided into a training set $\mathcal{D}_i^{\text{tr}}$ and a testing set $\mathcal{D}_i^{\text{ts}}$, i.e., $\mathcal{T}_i = \{(X_i^1, y_i^1), \ldots, (X_i^{N_t}, y_i^{N_t})\} = \{\mathcal{D}_i^{tr}, \mathcal{D}_i^{ts}\}$.

Problem Definition. For a specific target grid cell s, given all the historical traffic data, we aim to predict the traffic status $\{\hat{y}^t\}$ in consecutive time slots based on the available traffic-related features $\{X^t\}$. Since our goal is using meta-learning to solve this problem, the problem is transformed as follows:
In meta-learning setup, the historical time series traffic data is segmented into τ tasks; we assume all the tasks are sampled from the same distribution, $\mathcal{T}_i \sim p(\mathcal{T})$. During meta-training, we aim to train a meta-learner (with parameters θ) whose objective is to minimize the expected loss with respect to θ over all training tasks sampled from $p(\mathcal{T})$:

$$\theta^\star = \arg\min_\theta \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}\left(\phi_i, \mathcal{D}_i^{\text{ts}}\right), \text{and } \phi_i = f_\theta\left(\mathcal{D}_i^{\text{tr}}\right). \tag{1}$$

During meta-testing, the meta-learner is evaluated on unseen testing tasks from the same task distribution. When predicting the future traffic, which can be view as a new testing task, we have:

$$\hat{y}^t = f_{\theta\star}(\mathcal{D}^{\text{tr}}, X^t), \tag{2}$$

where $\mathcal{D}^{\text{tr}} = \{(X^1, y^1), \ldots, (X^{t-1}, y^{t-1})\}$ includes a few training data in the current task. The problem is illustrated in Figure 2.

2.2 Solution Framework

Figure 3 shows the solution framework. All the historical traffic data including traffic status and traffic-related features is segmented into different small tasks. cST-ML is modeled based on Bayesian black-box meta-learning framework combined with novel designs that help to capture traffic uncertainties and spatial-temporal dependencies. During meta-training, the cST-ML is applied to each meta-training task to perform traffic prediction, the parameters of cST-ML are updated based on the predicted loss. The well-trained cST-ML can be fast adapted to any new traffic prediction tasks and exhibit excellent performance during meta-testing time. The detailed data preparation and task segmentation process will be presented in Section IV. We will first introduce the methodologies in the next section.

3 METHODOLOGIES

In this section, we detail the key challenges of the urban dynamics prediction problem, and introduce our continuous spatial-temporal meta-learning framework.

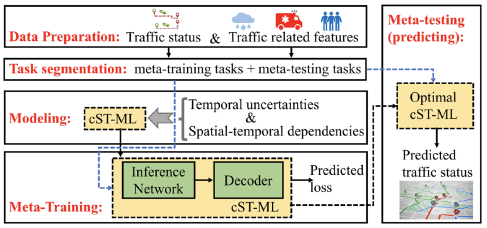


Fig. 3. Insight of the framework.

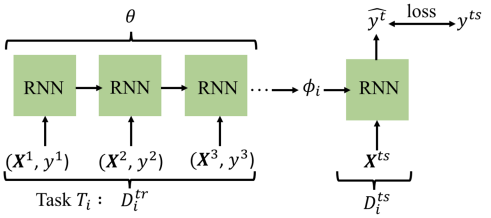


Fig. 4. Deterministic black-box meta-learning.

3.1 Key Challenges

State-of-the-art Meta-learning. The goal of meta-learning is to train a model that can quickly adapt to a new task using only a few data points. To accomplish this, the meta-learner f_θ is trained during a meta-training process on a set of training tasks that are sampled from the same task distribution, i.e., $\mathcal{T}_i \sim p(\mathcal{T})$, such that the trained meta-learner can quickly adapt to new unseen tasks using only a small number of examples. In effect, the meta-learning problem treats entire tasks as training examples and it is a generalization across tasks rather than across data points. For each task $\mathcal{T}_i$, there are two sets of data $\mathcal{D}_i^{\text{tr}}$ and $\mathcal{D}_i^{\text{ts}}$, where $\mathcal{D}_i^{\text{tr}}$ is for task adaption and getting task specific parameters ϕ_i, $\mathcal{D}_i^{\text{ts}}$ is used for calculating the loss and updating meta-learner parameters θ. The deterministic black-box meta-learner's objective is the same as Equation (1), where the loss function $\mathcal{L}(\phi_i, \mathcal{D}_i^{\text{ts}})$ can be mean-squared error.

The deterministic black-box meta-learning framework is illustrated in Figure 4. The common structure of black-box meta-learning is **recurrent neural network (RNN)**-based, where for task $\mathcal{T}_i$, the parameters of RNN can be viewed as θ, the hidden state ϕ_i is the adapted parameters for the current task, and the last cell of RNN is used for testing. Thus, the distribution $q(\phi_i|\mathcal{D}_i^{\text{tr}}, \theta)$ is deterministic in this setup, which means there is no uncertainties in $\mathcal{D}_i^{\text{tr}}$, i.e., $\phi_i = f_\theta(\mathcal{D}_i^{\text{tr}})$. However, in traffic dynamics prediction problem, even though the tasks are segmented based on time (e.g., everyday traffic is a task), there still exist temporal uncertainties and dynamics for each task, thus, deterministic black-box meta-learning is not enough when dealing with the traffic dynamics prediction problem.

Therefore, Bayesian black-box meta-learning is further developed to capture the task uncertainties, its objective is to maximize the log likelihood lower bound across all meta-training tasks:

$$\max_\theta \mathbb{E}_{\mathcal{T}_i}\Big[\mathbb{E}_{q(\phi_i|\mathcal{D}_i^{\text{tr}},\theta)}\Big[\log p\left(y_i^{\text{ts}}|\boldsymbol{X}_i^{\text{ts}}, \phi_i\right)\Big] - D_{KL}\left(q\left(\phi_i|\mathcal{D}_i^{\text{tr}}, \theta\right)\|p\left(\phi_i|\theta\right)\right)\Big], \tag{3}$$

where q is the inference network and parameterizes the mean and log-variance diagonal of a Gaussian distribution, and ϕ_i is sampled from this distribution, which indicates the latent distribution of ϕ_i incorporates the uncertainties of tasks during each adaptation process. Thus, the Bayesian black-box meta-learning can capture temporal uncertainties and dynamics of tasks by maximizing the variational lower bound across all meta-training tasks.

Challenges. The Bayesian black-box meta-learning only captures the uncertainties among different tasks, it does not consider the complex traffic spatial-temporal dependencies and temporal uncertainties within tasks. Thus, in traffic dynamics prediction, we need to incorporate the spatial-temporal dependencies and temporal uncertainties within tasks into the Bayesian black-box meta-learning framework and design unique structures for meta-learner to tackle these challenges.

3.2 cST-ML Modeling

Considering all the challenges highlighted above, it is not enough to only employ the Bayesian black-box meta-learning framework, which just takes temporal uncertainties among tasks into account, we also need to design unique structures for the meta-learner to tackle the complex spatial and temporal traffic dependencies and temporal uncertainties within tasks. Thus, now we are in a position to develop our cST-ML framework.

Following the original Bayesian black-box meta-learning, to deal with the uncertainties in task adaptation, each time we sample the adapted parameters from a latent distribution and thus the adapted parameters can be viewed as a latent variable. Our goal is still to maximize the log likelihood across all meta-training tasks, and we approximate the likelihood with **variational lower bound (ELBO)** [6]. The ELBO is derived as Equation (4):

$$\begin{aligned} \log p(x) &\geq \mathbb{E}_{q(z|x)}[\log p(x,z)] + H(q(z|x)) \\ &= \mathbb{E}_{q(z|x)}[\log p(x|z)] - D_{KL}(q(z|x)\|p(z)), \end{aligned} \tag{4}$$

where z is the latent variable and x is the real data, D_{KL} is the Kullback-Leibler divergence, $p(x|z)$ can be treated as an decoder and $q(z|x)$ is the inference network, $p(z) \sim N(0,1)$.

In traffic dynamics prediction, to advance the Bayesian black-box meta-learning framework and take uncertainties within tasks into consideration, we first segment the historical traffic data into τ tasks, for each task, instead of directly dividing the current task into $\mathcal{D}_i^{\text{tr}}$ and $\mathcal{D}_i^{\text{ts}}$ and applying cST-ML only once, we slide cST-ML as a rolling window within the task. Just as illustrated in Figure 5(b). The rolling windows capture the inner temporal uncertainties within tasks and thus help to improve the prediction accuracy.

In this situation, for task $\mathcal{T}_i$ and its jth rolling window, we have specific $\mathcal{D}_{i,j}^{\text{tr}}$ and $\mathcal{D}_{i,j}^{\text{ts}}$. Equation (5) is the log likelihood lower bound of the jth rolling window in task $\mathcal{T}_i$:

$$\begin{aligned} L_\theta\left(\phi_{i,j}, \mathcal{D}_{i,j}^{\text{ts}}\right) = \mathbb{E}_{q(\phi_{i,j}|\mathcal{D}_{i,j}^{\text{tr}},\theta)}\left[\log p\left(y_{i,j}^{\text{ts}}|\boldsymbol{X}_{i,j}^{\text{ts}}, \phi_{i,j}\right)\right] \\ -D_{KL}\left(q\left(\phi_{i,j}|\mathcal{D}_{i,j}^{\text{tr}},\theta\right)\|p(\phi_{i,j}|\theta)\right), \end{aligned} \tag{5}$$

where q is the inference network and parameterizes the mean and log-variance diagonal of a Gaussian distribution, and $\phi_{i,j}$ is sampled from this distribution for each rolling window, the Kullback-Leibler divergence can be approximated using the reparameterization trick (see more information in Reference [13]). Compared with Equation (4), the latent variable corresponds to the adapted parameter $\phi_{i,j}$, and the information we use to infer $\phi_{i,j}$ includes $\mathcal{D}_{i,j}^{\text{tr}}$ and θ.

The log likelihood lower bound of all rolling windows within task $\mathcal{T}_i$ is presented in Equation (6):

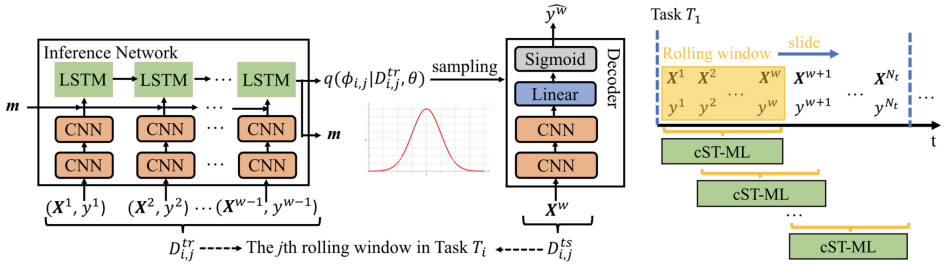
$$L_\theta(\mathcal{T}_i) = \sum_j L_\theta\left(\phi_{i,j}, \mathcal{D}_{i,j}^{\text{ts}}\right). \tag{6}$$

Thus, in traffic dynamics prediction problem, the final objective is to maximize the log likelihood lower bound across all meta-training tasks:

$$\max_\theta \mathbb{E}_{\mathcal{T}_i}[L_\theta(\mathcal{T}_i)]. \tag{7}$$

3.3 cST-ML Architecture

We also design unique structures for cST-ML to tackle the complex spatial-temporal traffic dependencies. The structure of our cST-ML is composed of an inference network and a decoder. The inference network tries to encode the training data within a task into a latent distribution, which captures the spatial patterns of the current location and also learns the temporal dependencies and



(a) cST-ML structure (b) Rolling window illustration

Fig. 5. cST-ML performs as a rolling window.

ALGORITHM 1: Meta-training

Input: Task distribution $p(\mathcal{T})$, window size w, step size $c = 1$, initialized cST-ML f_{θ_0}.
Output: Well trained cST-ML.

```
while not done do
    Sample a task T_i ~ p(T).
    Prepare D^tr_{i,j} and D^ts_{i,j} for each rolling window in T_i
    for all rolling windows in T_i do
        Sample φ_{i,j} from q(φ_{i,j} | D^tr_{i,j}, θ).
        Compute log likelihood using Equation (5).
    end for
    Update θ with Adam [12] to maximize Equation (6).
end while
```

uncertainties, the decoder is responsible for the prediction using the testing data within the same task. Figure 5 shows the overall structure of cST-ML.

The Inference Network is CNN- and LSTM-based and is actually the adaptation process of a task, which takes in the $\mathcal{D}^{\text{tr}}_{i,j}$ and extracts information from $\mathcal{D}^{\text{tr}}_{i,j}$, aiming to output a latent distribution, which captures uncertainties of the jth rolling window in $\mathcal{T}_i$. The input of the inference network includes two parts, (i) $\mathcal{D}^{\text{tr}}_{i,j} = \{(X^1_i, y^1_i), \ldots, (X^t_i, y^t_i)\}$, where $t < N_t$, and (ii) a vector $\boldsymbol{m}$ containing memories from the previous rolling windows within the current task $\mathcal{T}_i$, which is actually the hidden state of the LSTM in the last time step. In the first rolling window of $\mathcal{T}_i$, the input memory is a zero vector.

Since $\boldsymbol{X}^t$ is a tensor for each time slot, y^t is first enlarged to an $\ell \times \ell$ matrix and concatenates with $\boldsymbol{X}^t$, and then the concatenated tensor goes through a few layers of CNN activated by ReLU, which can capture the spatial dependencies of local traffic. The output sequence then concatenates with the memory vector and becomes the input of the LSTM, the hidden state of LSTM in the last time slot t goes through fully connected layers and produces the mean and log variance of a Gaussian distribution $q(\phi_{i,j}|\mathcal{D}^{\text{tr}}_{i,j}, \theta)$.

The Decoder aims to produce the prediction $\hat{y}^{ts}$ based on $\boldsymbol{X}^{ts}$ where $(\boldsymbol{X}^{ts}, y^{ts}) \in \mathcal{D}^{\text{ts}}_{i,j}$, the prediction loss is calculated using y^{ts} and $\hat{y}^{ts}$. Decoder takes two inputs, (i) the adapted information $\phi_{i,j}$ sampled from $q(\phi_{i,j}|\mathcal{D}^{\text{tr}}_{i,j}, \theta)$ and (ii) $\boldsymbol{X}^{ts}$. $\boldsymbol{X}^{ts}$ first goes through a few layers of CNN activated by ReLU and then concatenates with $\phi_{i,j}$, the results pass fully connected layers activated by Sigmoid function, and we get the final prediction $\hat{y}^{ts}$. The detailed structure of the inference network and decoder are illustrated in Figure 5(a).

ALGORITHM 2: Meta-testing

Input: A new task $\mathcal{T} = \{(\boldsymbol{X}^1, y^1), \ldots, (\boldsymbol{X}^{t-1}, y^{t-1})\}$ with available $\boldsymbol{X}^t, \ldots, \boldsymbol{X}^{N_t}$, window size $w = t$, step size $c = 1$, well-trained cST-ML $f_{\theta^\star}$.
Output: Predicted values $\{\hat{y}^t, \ldots, \hat{y}^{N_t}\}$.
1: Define $\mathcal{D}^{\text{tr}} = \{(\boldsymbol{X}^1, y^1), \ldots, (\boldsymbol{X}^{t-1}, y^{t-1})\}$ and $\mathcal{D}^{\text{ts}} = \{\boldsymbol{X}^t\}$ as the first rolling window in $\mathcal{T}$
2: **for** all rolling windows in $\mathcal{T}$ **do**
3: $\hat{y}^t = f_{\theta^\star}(\mathcal{D}^{\text{tr}}, \mathcal{D}^{\text{ts}})$.
4: Update $\mathcal{D}^{\text{tr}} = \{(\boldsymbol{X}^2, y^2), \ldots, (\boldsymbol{X}^t, \hat{y}^t)\}$ and $\mathcal{D}^{\text{ts}} = \{\boldsymbol{X}^{t+1}\}$ for the next rolling window.
5: **end for**

3.4 cST-ML Training and Testing

Since we perform cST-ML as a rolling window within tasks, assume the rolling window size is w and step size is 1, for the 1st rolling window, we use the first $w-1$ data points $\{(\boldsymbol{X}_i^1, y_i^1), \ldots, (\boldsymbol{X}_i^{w-1}, y_i^{w-1})\}$ in $\mathcal{T}_i$ as input of the inference network and use $\boldsymbol{X}_i^w$ as the input of decoder to predict $\hat{y}_i^w$, thus, $\mathcal{D}_{i,j}^{\text{tr}} = \{(\boldsymbol{X}_i^1, y_i^1), \ldots, (\boldsymbol{X}_i^{w-1}, y_i^{w-1})\}$ and $\mathcal{D}_{i,j}^{\text{ts}} = \{(\boldsymbol{X}_i^w, y_i^w)\}$, just as illustrated in Figure 5. Then, we use data points $\{(\boldsymbol{X}_i^2, y_i^2), \ldots, (\boldsymbol{X}_i^w, y_i^w)\}$ as input of inference network and use $\boldsymbol{X}^{w+1}$ as the input of decoder to predict $\hat{y}^{w+1}$ and so on so forth. In this situation, for task $\mathcal{T}_i$ and its jth rolling window, we have specific $\mathcal{D}_{i,j}^{\text{tr}}$ and $\mathcal{D}_{i,j}^{\text{ts}}$, and we backpropagate through the total loss of all rolling windows in task $\mathcal{T}_i$ to update meta-learner θ (i.e., the parameters of both inference network and decoder).

The detailed meta-training process is shown in Algorithm 1. We repeatedly sample tasks from the task distribution, for each sampled task, we compute the total log likelihood for all rolling windows and update θ once.

After training, the well-trained meta-learner θ can fast adapt to any new tasks. The meta-testing algorithm is shown in Algorithm 2. In meta-testing, to predict the future traffic, we define a new testing task $\mathcal{T} = \{(\boldsymbol{X}^1, y^1), \ldots, (\boldsymbol{X}^{t-1}, y^{t-1})\}$, after the prediction of $\hat{y}^t$, we update the $\mathcal{D}^{\text{tr}}$ and $\mathcal{D}^{\text{ts}}$ of the current rolling window and slide the window to get more predictions.

4 EVALUATION

In this section, we conduct extensive experiments on real-world traffic datasets to evaluate our cST-ML. We first describe the datasets and introduce experiments, then we present baselines compared with our model and the evaluation metrics. Finally, the experiment results are presented and analyzed in detail.

4.1 Dataset Descriptions

Preprocessing of Dataset. We evaluate our model on the real-world datasets including (1) traffic speed, (2) taxi inflow, and (3) travel demand, all of which are extracted from Shenzhen, China, from July 1 to December 31, 2016. In the preprocessing step, we first apply map gridding to the whole Shenzhen City, where the city is partitioned into 40×50 grid cells, for each target grid cell, its target region is the 5×5 matrix with the target grid cell in center. Thus, there are in total 1,656 possible target grid cells. The map gridding method, the target grid cells and its corresponding target regions are illustrated in Figure 6.

Traffic speed, taxi inflow and travel demand are all extracted from taxi GPS records collected in Shenzhen, China, from July 1 to December 31, 2016. In each time slot (i.e., 1 h) of each day, taxi inflow is the number of taxis that stay or arrive at a target grid cell, travel demand is the number

(a) Road map gridding (b) Target grid cells and regions

Fig. 6. Map gridding and target grid cells illustration.

of taxi pickups within a target grid cell. In effect, it is hard to obtain the travel demands of all transport modes in a target grid cell, thus, we use taxi demands to represent travel demands.
Experiment Descriptions. Next, we describe our two traffic prediction experiments we will perform in detail.

- **Traffic speed prediction.** In speed prediction, the traffic status in each grid cell is measured by average traffic speed, and there are 12 time slots per task, i.e., $N_t = 12$, and thus 184 tasks over 6 months. All the tasks are divided into meta-training tasks (the first 80% of all tasks) and meta-testing tasks (the rest of 20%). We treat travel demands, traffic inflow and the time of the day as traffic-related features, and use meta-training tasks to train the model and use meta-testing tasks to do evaluations. The goal of this task is to predict the traffic speed of a target grid cell s based on the historical available features.
- **Taxi inflow prediction.** Similar to traffic speed prediction, in the taxi inflow prediction, the traffic status in each grid cell is measured by taxi inflow. We view travel demands, traffic speed and the time of the day as traffic-related features. There are also 12 time slots per task, i.e., $N_t = 12$, and 184 tasks over 6 months. All the tasks are divided into meta-training tasks (the first 80% of all tasks) and meta-testing tasks (the rest of 20%). We aim to train the model with meta-training tasks and evaluate the model using meta-testing tasks.

4.2 Baselines

- **HA [24].** For each grid cell, **Historical Average (HA)** method predicts the traffic status for a target grid cell based on its average status of the previous time slots.
- **Regression (Reg) [3].** This method applies ridge regression to predict the future traffic status, the predictors are the corresponding traffic-related features. The training data are used to train the regression model and the testing data are used for evaluations.
- **ARIMA [22]. Auto-Regressive Integrated Moving Average (ARIMA)** is a conventional parametric-based time-series model. Here, we view the historical traffic status as time-series data and apply ARIMA to predict the future traffic status.
- **CNN+LSTM [17, 32].** This method uses LSTM to predict the future traffic status using traffic-related features as input. The daily traffic-related features can be viewed as an input sequence, which goes through CNNs first and then passes LSTM to get the predicted traffic status sequence.
- **SNAIL [16].** It is a state-of-the-art deterministic black-box meta-learning method. SNAIL utilizes attention layers to get the deterministic adapted parameters for each task instead of sampling from a distribution, where the task uncertainties are not considered.
  cST-v1 [31]. It is the previous version of the cST-ML proposed in the conference paper, which does not have the memory vector and rolling windows.

Table 2. Performance on Traffic Speed Prediction and Taxi Inflow Prediction

Methods			HA	Reg	ARIMA	CNN+LSTM	SNAIL	cST-v1	cST-v2	**cST-ML**
Speed	1h	RMSE	2.993	2.224	2.160	2.923	2.216	0.981	1.237	**0.869**
		MAPE	0.170	0.124	0.124	0.156	0.126	0.075	0.085	**0.058**
	3h	RMSE	2.545	2.249	2.517	2.674	2.278	2.235	2.185	**2.119**
		MAPE	0.126	0.112	0.110	0.125	0.114	0.109	0.104	**0.093**
	6h	RMSE	3.120	3.035	3.711	3.000	2.933	3.360	2.955	**2.685**
		MAPE	0.197	0.183	0.221	0.199	0.171	0.207	0.179	**0.164**
Inflow	1h	RMSE	56.833	37.356	26.902	37.501	29.715	24.376	28.724	**16.461**
		MAPE	0.239	0.159	0.120	0.160	0.116	0.099	0.111	**0.061**
	3h	RMSE	65.929	38.572	35.551	38.940	31.191	27.364	24.429	**19.258**
		MAPE	0.237	0.145	0.119	0.147	0.097	0.078	0.075	**0.064**
	6h	RMSE	64.777	31.937	38.734	32.781	25.592	32.335	24.753	**18.744**
		MAPE	0.235	0.111	0.124	0.114	0.079	0.113	0.082	**0.061**

cST-v2 [31]. This method is similar to our proposed cST-ML; it has rolling windows but no memory module in the model.

4.3 Evaluation Metrics

We use **mean absolute percentage error (MAPE)** and **rooted mean-squared error (RMSE)** for evaluations:

$$\text{MAPE} = \frac{1}{T}\sum_{t=1}^{T} |y_t - \hat{y}_t| / y_t, \text{RMSE} = \sqrt{\frac{1}{T}\sum_{t=1}^{T} (y_t - \hat{y}_t)^2}, \tag{8}$$

where y_t is the ground-truth traffic status observed in the target grid cell s in the tth time slot, and $\hat{y}_t$ is the corresponding prediction, T is the total number of time slots to perform prediction.

4.4 Experimental Settings

Based on our final objective Equation (7), the goal is to maximize the log likelihood and negative KL divergence. In the implementation, since maximizing log likelihood equals to minimizing the mean-squared error of predictions, the objective can be transformed to minimizing the corresponding mean-squared error and KL divergence instead [2, 13].

In experiments, the whole Shenzhen city is divided into 40×50 grid cells with a side-length $l_1 = 0.0084°$ in latitude and $l_2 = 0.0126°$ in longitude. The target region for a target grid cell is of size 5×5, i.e., $\ell = 5$. Thus, there are in total 1,656 possible target grid cells in Shenzhen city. In the experiment, we can select any possible target grid cell to perform traffic predictions.

The time interval for each task used to train the cST-ML are from 7:00 a.m. to 7:00 p.m., where each hour is a time slot, and we have 12 time slots per day/task, i.e., $N_t = 12$. Thus, we have $\mathcal{T}_i = \{(\boldsymbol{X}_i^1, y_i^1), \ldots, (\boldsymbol{X}_i^{12}, y_i^{12})\}$.

The structure of cST-ML is as follows: two layers of CNN are utilized before LSTM in the inference network, the input channel of the first CNN is 4, the output channel is 64, the kernel size is 3, stride is 1, and padding is 1; for the second CNN layer, the input channel is 64, the output channel is 128, the kernel size is 5, stride is 1, and padding is 0. In decoder, we still use two layers of CNN combined with a linear transformation. cST-ML is trained using Adam optimizer [12] with $\beta_1 = 0.5$ and $\beta_2 = 0.999$, and a learning rate of 2×10^{-4} for 2,000 times task samplings, the window size is 5 with step size equal to 1.

4.5 Evaluation Results

4.5.1 Average Prediction Performance. First, we conduct experiments to compare the average prediction performance of our proposed cST-ML and competing baseline models. The results are shown in Table 2. In the table, for a specific target grid cell, we present the RMSE and MAPE

results for 1-h, 3-h, and 6-h traffic speed prediction and taxi inflow prediction. For meta-based models (including cST-ML and SNAIL), we randomly pick five meta-testing tasks in both of the traffic speed and taxi inflow predictions, compute the 1-h, 3-h, and 6-h RMSE and MAPE for each testing task and report the average results in the table. For other models, we use the same testing data to compute the statistics.

In traffic speed prediction, according to the average RMSE and MAPE in 1-h, 3-h, and 6-h predictions, cST-ML outperforms all the baseline models. Compared with cST-v1 and cST-v2, our updated cST-ML in Table 2 can predict traffic status more accurately in longer time slots and always has great improvements in both metrics. The reason is that in our updated cST-ML, the memory vector can help to capture the temporal dependencies of traffic within each task, and the rolling windows can better capture the temporal uncertainties, both of which lead to better prediction performance.

SNAIL is a deterministic black-box meta-learning method that does not consider any uncertainties of tasks, but it achieves competitive performance in some cases (i.e., 6-h traffic speed and taxi inflow predictions), the reason is that we view daily traffic as one task in meta-training and meta-testing, for one specific target grid cell, in most of cases, the everyday traffic is similar, which means there is less task uncertainties, and thus SNAIL can achieve competing prediction performance sometimes.

CNN+LSTM is used as a seq2seq model in traffic prediction, which utilizes the traffic-related features to predict the traffic status, so it does not rely on the previous traffic status in testing or prediction process, which could result in larger prediction errors.

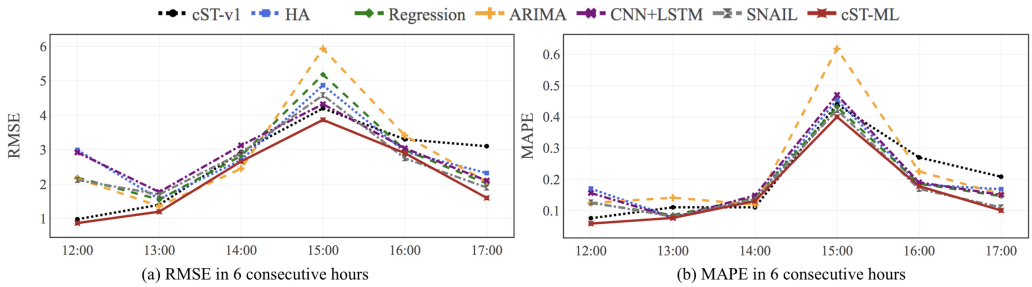
Compared with the traditional traffic prediction baseline models including HA, Regression and ARIMA, cST-ML achieves significant improvements, since it not only captures the traffic spatial-temporal dependencies but also the temporal uncertainties. On the contrary, these traditional models only consider either the temporal dependencies or the relationships between traffic status and features, and they cannot deal with the traffic uncertainties very well.

In taxi inflow prediction, we get similar prediction results. SNAIL is the most competitive baselines compared with other baseline models, which indicates they can better learns the spatial-temporal patterns of traffic, and thus obtain lower errors. However, cST-ML is more powerful due to its novel design.

4.5.2 Detailed Performance in Consecutive Time Slots. In this part, we are aiming to prove the effectiveness of our cST-ML in traffic predictions in each time slot (e.g., 1 h). In urban traffic prediction problem, the good average prediction performance is not enough, since we expect to get more accurate prediction for each specific time slot. Thus, we conduct experiments and provide detailed prediction performance for each time slot (i.e., 1 h). The statistics are calculated based on five meta-testing tasks in both of the traffic speed and taxi inflow predictions, in each time slot, we report the average RMSE and MAPE of all five testing tasks.

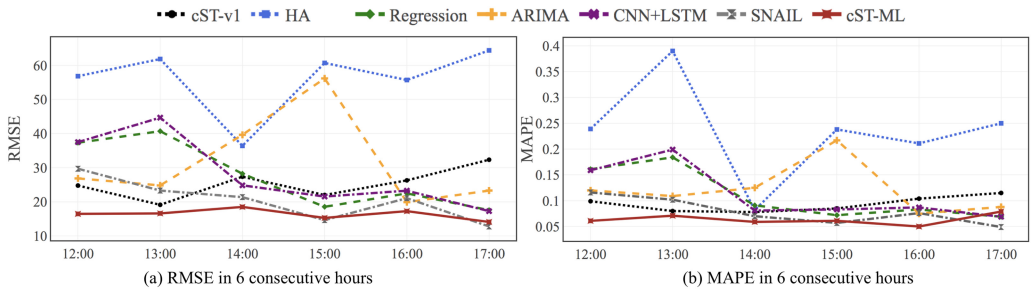
In traffic speed prediction, the detailed performance is presented in Figure 7. As shown in Figures 7(a) and 7(b), our cST-ML achieves the best prediction performance in most of the time slots, but in some cases, some baseline models would have slightly better performance, for example, ARIMA has the best performance at 14:00 and SNAIL has the best performance at 16:00. However, the performance of baselines including ARIMA and SNAIL presents higher volatilities and thus the prediction performance is much more unstable.

In taxi inflow prediction, as shown in Figures 8(a) and 8(b), our cST-ML also achieves the best prediction performance except in a few time slots, for example, SNAIL has slightly better performance than cST-ML at 14:00 hour and 17:00, respectively. However, similar to Figure 7, the performance of all baseline models still presents much higher volatilities in prediction performance, in contrast, cST-ML displays more stable and accurate predictions in general, which also proves that cST-ML



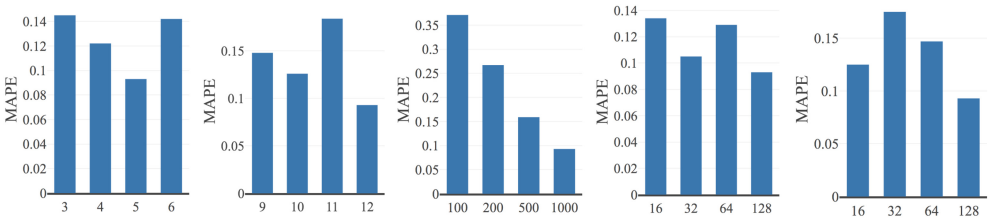
(a) RMSE in 6 consecutive hours (b) MAPE in 6 consecutive hours

Fig. 7. Comparisons of models in 6 consecutive hours in traffic speed prediction.



(a) RMSE in 6 consecutive hours (b) MAPE in 6 consecutive hours

Fig. 8. Comparisons of models in 6 consecutive hours in taxi inflow prediction.



(a) Rolling window size (b) Task length (c) Number of task iterations in training (d) Dimension of memory vector and hidden states (e) Dimension of μ and σ

Fig. 9. Impact of parameters in traffic speed prediction.

can better capture the traffic uncertainties and complex spatial-temporal dependencies, therefore, cST-ML provides more accurate and stable traffic prediction in consecutive time slots.

4.5.3 Evaluations on cST-ML Parameters. cST-ML has many hyper-parameters, e.g., rolling window size, task length, and so on. In this part, we conduct experiments to evaluate the impacts of different hyper-parameters on our cST-ML. In Figures 9 and 10, the experimental results are presented to demonstrate how different values of hyper-parameters influence the performance of cST-ML. Specifically, the hyper-parameters we aim to analyze includes rolling window size, task length, the number of training iterations, the dimension of hidden states in LSTM (inside the inference network of cST-ML) and the dimension of mean and log variance, which are used to define the output distribution of inference network in cST-ML. All the statistics in Figures 9 and 10 are MAPEs of 3-h predictions, which are computed using 5 meta-testing tasks in both of traffic speed prediction and taxi inflow prediction.

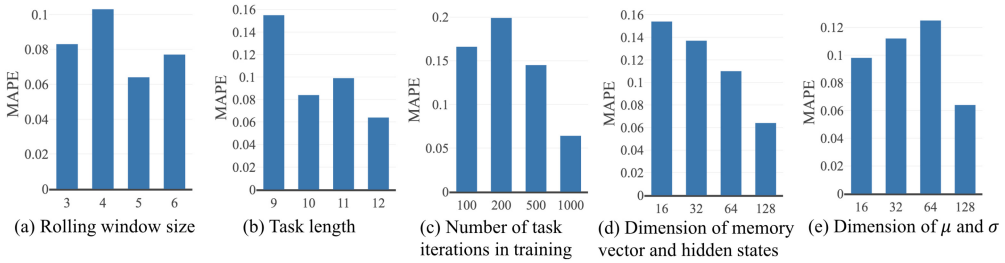


Fig. 10. Impact of parameters in taxi inflow prediction.

As shown in Figures 9(a) and 10(a), the prediction performance is sensitive to rolling window size. The errors are both high when the rolling window size is too small or large. When the rolling window size is small (e.g., window size equal to 3 or 4), we only use a few data points (e.g., two or three data points) within a task to do the next step prediction, where little traffic information is provided in task adaptation and thus leads to high prediction errors. However, if the the rolling window size is too large, the temporal uncertainties are less captured, when the rolling window size equal to the task length, no temporal uncertainties within a task can be captured, which also lead to poor prediction performance.

In Figures 9(b) and 10(b), task length also influences the performance of our model and when task length is equal to 12, we have the best performance. Since the historical traffic data (from 7:00 a.m. to 7:00 p.m. every day) can be viewed as time series data, if we segment the tasks by 12 h, each task contains complete traffic information of one day. In general, everyday traffic patterns are similar, the assumption that all tasks are sampled from the same distribution is satisfied in meta-learning framework, which leads to higher prediction accuracy. On the contrary, if the tasks are not segmented by 12 h, tasks may display greatly different traffic patterns where the meta-learning assumption is hard to be satisfied and thus usually results in high prediction errors.

In Figures 9(c) and 10(c), we can easily conclude that the more training iterations we have, the lower prediction errors cST-ML produces. Since in each training iteration, we randomly sample a task from the task distribution, the more times we sample, the better meta-knowledge we get through the whole training process, which certainly will lead to better performance.

Next, we treat the dimension of the memory vector as one hyperparameter and conduct experiments to demonstrate how different memory dimensions influence the performance. The results are shown in Figures 9(d) and 10(d). Since the dimension of memory vector is the same as the dimension of hidden states in LSTM (within our cST-ML) in our implementation, Figures 9(d) and 10(d) indicate both impacts of memory dimension and hidden states dimension. We find that the model performance is very sensitive to this dimension. A higher dimensionality leads to better performances, which indicates more spatial and temporal information being kept by LSTM. At the meantime, more detailed memories are also kept by the model.

Similarly, we analyze how the dimension of mean and log variance influence performance in Figures 9(e) and 10(e). Mean and log variance are the outputs of inference network in cST-ML, by which the distribution of adapted parameters is determined. We find that higher dimension leads to lower error, which indicates more information of the output distribution is captured in a task.

4.6 Case Studies

To further illustrate the effectiveness of our cST-ML to capture traffic dynamics, we perform a case study in this subsection. Since different locations could show different traffic patterns, e.g., in Figure 11(a), two representative target grid cells G1 and G2 are presented, the target region of

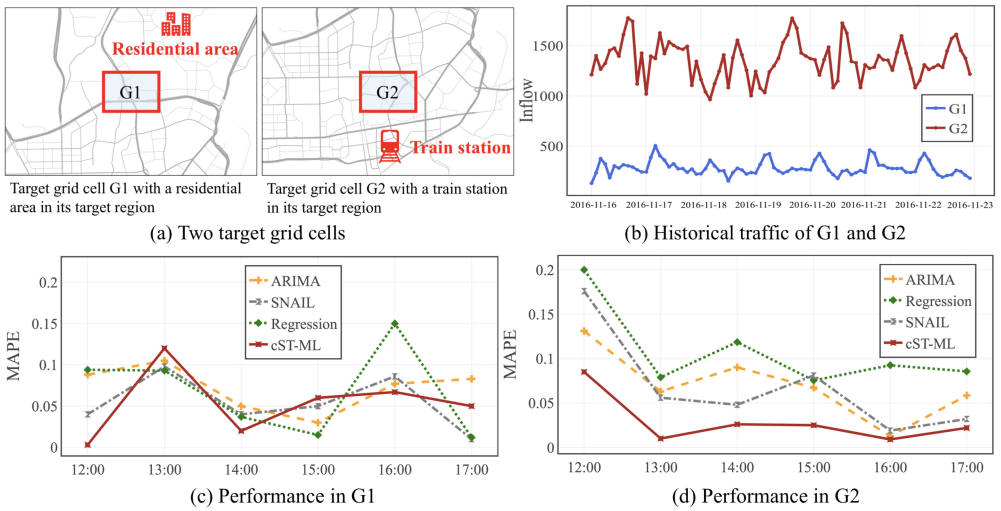


Fig. 11. Traffic predictions of two target grid cells.

G1 contains a residential area and the target grid cell G2 is close to a train station. As shown in Figure 11(b), the historical traffic (i.e., 7 days taxi inflow) are plotted, where the taxi inflow of G2 presents obvious fluctuations and no regular patterns can be captured, since the travel demand in G2 varies every day, while the daily traffic patterns of G1 are more consistent due to the similar daily travel demands in this area. In this case study, we study the traffic prediction performance of our cST-ML when dealing with such two different target grid cells. We compare our cST-ML with three competitive baselines and the performance is shown in Figures 11(c) and 11(d). When performing traffic prediction in a location with obvious temporal dynamics such as G2, cST-ML outperforms all other baselines, while in a location with consistent traffic patterns like G1, some baselines can provide reasonable predictions as well. This case demonstrates that our cST-ML has excellent capability to deal with traffic dynamics and temporal uncertainties and thus tends to present better performance when local traffic presents greater fluctuations and irregular patterns.

4.7 Discussions

There are still some important problems about our model and our experiment deserve to be discussed:

- **Influences of different time dynamics.** In the previous experiments, each task (i.e., one day) contains 12 time slots, and each time slot is 1 h. In this part, we want to discuss how the length of time slots (i.e., different time dynamics) would influence the performance of our cST-ML. We compare the cST-ML performance of traffic speed prediction when the time dynamics is 0.5 h, 1 h, and 2 h, the prediction performance of cST-ML on next three time slots is presented in Figure 12. We can find when the time dynamics is 1 h, we have the best performance. When the time dynamics is 0.5 h or 2 h, cST-ML presents higher RMSE, since more fine-grained time dynamics (e.g., 0.5-h dynamics) usually present higher temporal uncertainties and thus harder to predict, by contrast, longer time dynamics (e.g., 2-h dynamics) lead to fewer training samples in each task and thus result in low training quality and bad performance.
- **Capturing weekly or seasonal traffic dynamics patterns.** In this article, we divide the whole time-series traffic data into different tasks, since everyday traffic usually shares a lot

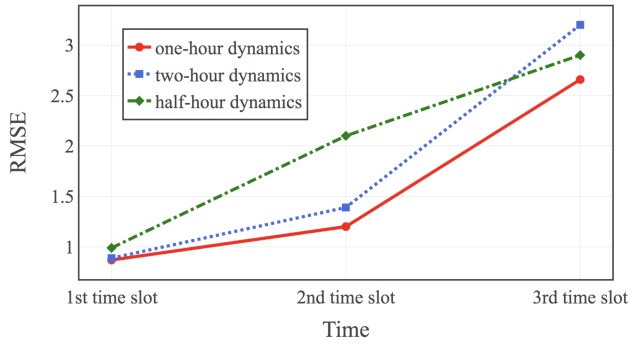


Fig. 12. cST-ML performance of traffic speed prediction on different time dynamics.

in common, we view one day traffic as a task. Besides, it is feasible for the cST-ML to capture weekly or seasonal traffic patterns as long as enough traffic data is provided. For example, we can also treat weekly or seasonal data as different tasks, but to ensure the good training quality, we need a lot more weekly or seasonal tasks for training and thus a larger dataset is needed. However, due to the limitation of our dataset size, which only contains six months traffic data, it is improper to apply weekly or seasonal data as tasks in the experiments.

- **Model adaptation across cities.** The well-trained meta-learner can be adapted across cities as long as a large amount of data of different cities are used for meta-training. In meta-learning settings, the meta-training tasks can be any similar tasks, which indicates the daily traffic data of different locations and cities can be viewed as different tasks and used for meta-training. Thus, if the meta-learner is trained with traffic data of multiple cities, during meta-testing, given a few hours traffic data of any location in any city, the well-trained meta-learner can output the future traffic.
- **Predicting multiple traffic dynamics.** Our proposed meta-learner can be used to predict multiple urban dynamics, such as traffic inflow, speed, and so on. If various traffic data is prepared for meta-training, then the traffic status y^t in Definition 4 in Section 2 will be measured with different metrics (e.g., traffic inflow, traffic speed, etc.). And in each traffic prediction task $\mathcal{T}_i = \{(\boldsymbol{X}_i^1, y_i^1), \ldots, (\boldsymbol{X}_i^{N_t}, y_i^{N_t})\}$, y_i^t will be a vector instead of a single value. In the cST-ML architecture, y_i^t will pass CNN and LSTM in inference network, which helps to keep the correlations between different dynamics. And the objective function, training and testing algorithms will stay unchanged in this situation.
- **Influences of size of grid cells.** At last, in our experiments, the size of grid cells also matters. The traffic data (including traffic speed and taxi inflow) is extract from taxi GPS data in Shenzhen, China. The taxi GPS sensors send GPS information (including the current time, longitude and latitude) about every 10 to 40 s, if the size of grid cells is smaller, the extracted traffic speed and taxi inflow data will be less accurate and thus affect the prediction performance. If the size of grid cells is a lot larger, then the traffic data of each grid cell will present less dynamics, and such data is improper to evaluate our model, since our proposed cST-ML focuses on capturing the temporal dynamics of traffic data.

5 RELATED WORK

Urban Traffic Prediction. In urban traffic prediction area, some works focused on traffic volume and crowd flow prediction. For example, Reference [27] proposed a citywide traffic volume estimation framework that combined machine learning techniques and traffic flow theory. Another

work [23] developed novel real-time framework offering accurate arrival crowd flow prediction at subway stations. In addition, a lot of works adopt and advance the existing methods to predict urban traffic. For example, works such as References [5, 7, 21] applied support vector regression to predict future traffic and took environmental features into consideration. Reference [14] proposed a traffic prediction method that combined SARIMA model and autoregressive model with genetic algorithm optimization. Reference [28] tried to predict citywide flow using CNN, which better captured traffic spatial dependencies. References [15, 26] predict travel demands and traffic accidents using autoencoders and ConvLSTM, respectively. Other works, including References [8, 25], combine CNN and LSTM to predict the traffic speed and crowd flows. In our work, we aim to solve the urban traffic prediction problem using Bayesian meta-learning framework and capture traffic spatial-temporal dependencies and temporal uncertainties simultaneously.

Meta-Learning. A meta-learner is learned from training tasks and can be fast adapted into new tasks with just a few samples. The idea of meta-learning has been applied to many areas including supervised/unsupervised learning, reinforcement learning and even image generation. The state-of-the-art meta-learning methods including MAML [9], Reptile [19], SNAIL [16], MOCA [11], and so on. MAML and Reptile learn a good initialization of a model, which can be finetuned in new tasks. SNAIL is a black-box meta-learning method, where the black-box can be viewed as the meta-learner. They are not Bayesian meta-learning methods and do not consider any task uncertainties. MOCA augments a meta-learning algorithm with a differentiable Bayesian changepoint detection scheme, but it is not used to deal with time series predictions. In traffic prediction area, some works applied meta-learning methods to solve traffic prediction problem. For example, Reference [20] combined meta graph attention and meta recurrent neural network to capture spatial and temporal dependencies simultaneously. Another work [24] learned the meta-knowledge from multiple cities and performed the spatial-temporal traffic prediction. However, these two works still did not consider the task and temporal uncertainties in prediction.

6 CONCLUSION

In this article, we solved the traffic dynamics prediction problem using Bayesian meta-learning framework. We proposed a novel continuous spatial-temporal meta-learner (cST-ML), which learned a general traffic dynamics prediction strategy from historical traffic data (segmented into tasks) and could be quickly adapted to new prediction tasks containing just a few samples and exhibited excellent prediction performance. cST-ML captured the traffic spatial-temporal dependencies and the traffic uncertainties through new features in both objective and architecture beyond the original Bayesian black-box meta-learning. Novel training and testing algorithms were also designed for cST-ML where the traffic temporal uncertainties and dynamics were better kept by rolling windows. We conduct experiments on real-world traffic datasets (taxi inflow and traffic speed) to evaluate our proposed cST-ML. The experiment results verify that cST-ML can significantly improve the urban traffic prediction performance especially when obvious traffic uncertainties are presented and significantly outperforms all baseline models.

REFERENCES

[1] Github. 2020. cST-ML. Retrieved from https://github.com/cST-ML/cST-ML.

[2] Github. 2020. VAE pyTorch Tutorial. Retrieved from https://github.com/Jackson-Kang/Pytorch-VAE-tutorial.

[3] Ishteaque Alam, Dewan Md. Farid, and Rosaldo J. F. Rossetti. 2019. The prediction of traffic flow with regression analysis. In *Proceedings of the International Conference on Emerging Technologies in Data Mining and Information Security (IEMIS'19)*. 661–671.

[4] Han Bao, Xun Zhou, Yingxue Zhang, Yanhua Li, and Yiqun Xie. 2020. COVID-GAN: Estimating human mobility responses to COVID-19 pandemic through spatio-temporal conditional generative adversarial networks. In *Proceedings of the 28th International Conference on Advances in Geographic Information Systems (SIGSPATIAL'20)*. 273–282.

[5] Manoel Castro-Neto, Young-Seon Jeong, Myong-Kee Jeong, and Lee Han. 2009. Online-SVR for short-term traffic flow prediction under typical and atypical traffic conditions. *Expert Syst. Appl.* 36 (2009), 6164–6173.
[6] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In *Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS'16)*.
[7] Yuliang Cong, Jianwei Wang, and Xiaolei Li. 2016. Traffic flow forecasting by a least squares support vector machine with a fruit fly optimization algorithm. *Procedia Eng.* 137 (2016), 59–68.
[8] Zhiyong Cui, Ruimin Ke, and Yinhai Wang. 2017. Deep stacked bidirectional and unidirectional LSTM recurrent neural network for network-wide traffic speed prediction. In *Proceedings of the 6th International Workshop on Urban Computing*.
[9] 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.
[10] Eric J. Gonzales, Ci (Jesse) Yang, Ender Faruk Morgul, and Kaan Ozbay. 2014. *Modeling Taxi Demand with GPS Data from Taxis and Transit.* Technical Report. Mineta National Transit Research Consortium.
[11] James Harrison, Apoorva Sharma, Chelsea Finn, and Marco Pavone. 2020. Continuous meta-learning without tasks. In *Advances in Neural Information Processing Systems*, Vol. 33. Curran Associates, 17571–17581.
[12] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR'15)*.
[13] Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational Bayes. In *Proceedings of the International Conference on Learning Representations (ICLR'14)*.
[14] Xianglong Luo, Liyao Niu, and Shengrui Zhang. 2018. An algorithm for traffic flow prediction based on improved SARIMA and GA. *KSCE J. Civil Eng.* 22 (May 2018), 1–9.
[15] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, Fei-Yue Wang, et al. 2015. Traffic flow prediction with big data: A deep learning approach. *IEEE Trans. Intell. Transport. Syst.* 16, 2 (2015), 865–873.
[16] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. 2018. A simple neural attentive meta-learner. In *Proceedings of the International Conference on Learning Representations*.
[17] Olof Mogren. 2016. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. In *Proceedings of the Constructive Machine Learning Workshop (CML) at NIPS 2016*.
[18] Naoto Mukai and Naoto Yoden. 2012. Taxi demand forecasting based on taxi probe data by neural network. In *Proceedings of the International Conference on Intelligent Interactive Multimedia Systems and Services (IIMSS'12)*. 589–597.
[19] Alex Nichol, Joshua Achiam, and John Schulman. 2018. On first-order meta-learning algorithms. ArXiv, abs/1803.02999.
[20] 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 ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'19)*.
[21] Yuxing Sun, Biao Leng, and Guan Wei. 2015. A novel wavelet-SVM short-time passenger flow prediction in Beijing subway system. *Neurocomputing* 166 (Apr. 2015).
[22] M. Tan, S. C. Wong, J. Xu, Z. Guan, and P. Zhang. 2009. An aggregation approach to short-term traffic flow prediction. *IEEE Trans. Intell. Transport. Syst.* (2009), 60–69.
[23] Ermal Toto, Elke A. Rundensteiner, Yanhua Li, Richard Jordan, Mariya Ishutkina, Kajal Claypool, Jun Luo, and Fan Zhang. 2016. PULSE: A real time system for crowd flow prediction at metropolitan subway stations. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECMLPKDD'16)*.
[24] 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 *Proceedings of the World Wide Web Conference*. 2181–2191.
[25] 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).
[26] Zhuoning Yuan, Xun Zhou, and Tianbao Yang. 2018. Hetero-ConvLSTM: A deep learning approach to traffic accident prediction on heterogeneous spatio-temporal data. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'18)*. 984–992.
[27] Xianyuan Zhan, Yu Zheng, Xiuwen Yi, and Satish Ukkusuri. 2017. Citywide traffic volume estimation using trajectory data. *Trans. Knowl. Data Eng.* 29, 2 (2017), 272–285.
[28] Junbo Zhang, Yu Zheng, and Dekang Qi. 2016. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI'17)*. AAAI Press, 1655–1661.
[29] Yingxue Zhang, Yanhua Li, Xun Zhou, Xiangnan Kong, and Jun Luo. 2019. TrafficGAN: Off-deployment traffic estimation with traffic generative adversarial networks. In *Proceedings of the IEEE International Conference on Data Mining (ICDM'19)*.

[30] Yingxue Zhang, Yanhua Li, Xun Zhou, Xiangnan Kong, and Jun Luo. 2020. Curb-GAN: Conditional urban traffic estimation through spatio-temporal generative adversarial networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'20)*.

[31] Yingxue Zhang, Yanhua Li, Xun Zhou, and Jun Luo. 2020. cST-ML: Continuous spatial-temporal meta-learning for traffic dynamics prediction. In *Proceedings of the IEEE International Conference on Data Mining (ICDM'20)*.

[32] Jiachen Zhao, Fang Deng, Yeyun Cai, and Jie Chen. 2019. Long short-term memory—Fully connected (LSTM-FC) neural network for PM2.5 concentration prediction. *Chemosphere* (2019), 486–492. DOI: 10.1016/j.chemosphere.2018.12.128

[33] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. 2014. Urban computing: Concepts, methodologies, and applications. *ACM Trans. Intell. Syst. Technol.* 5, 3, Article 38 (September 2014), 55 pages. DOI: https://doi.org/10.1145/2629592

Received November 2020; revised April 2021; accepted July 2021