cST-ML: Continuous Spatial-Temporal Meta-Learning for Traffic Dynamics Prediction

Anonymous Author(s)

***Abstract*—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 paper, 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.**

***Index Terms*—traffic dynamics prediction, Bayesian meta-learning, spatial-temporal data.**

I. Introduction

Over 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, help to improve the efficiency of public transportation, but also guarantee the public safety [28].

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.

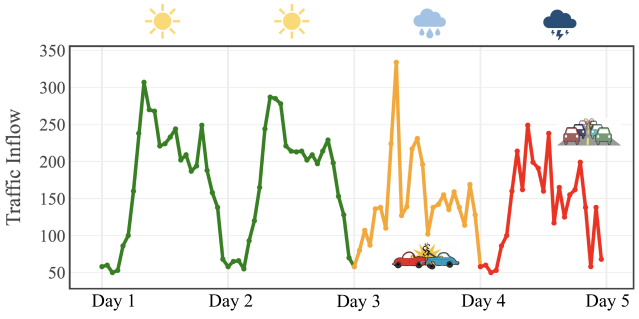


Fig. 1: Illustration of traffic dynamics.

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 use traditional machine learning methods and time series models to predict urban traffic. Works such as [3], [19] and [4] use support vector regression (SVR) to capture the relationships between traffic and environmental features. Another work [12] presents a traffic prediction method which combines the SARIMA model and multi-input autoregressive (AR) model with genetic algorithm (GA) optimization. In addition, deep neural networks are also widely used in urban traffic prediction works. For example, works [13] and [24] predict travel demands and traffic accidents using autoencoders and ConvLSTM, respectively. Other works including [23] and [5] combine CNN and LSTM to predict the traffic speed and crowd flows. However, these models 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. For example, a recent work [18] combines meta graph attention and meta recurrent

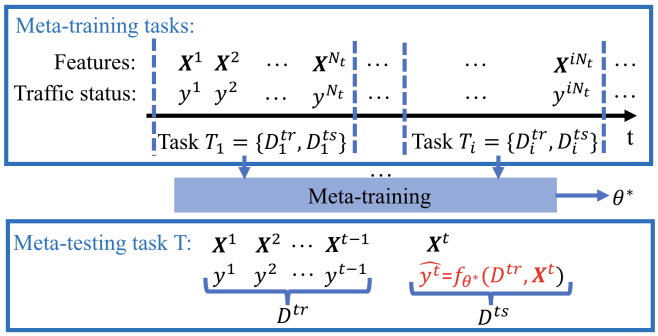


Fig. 2: Problem illustration.

neural network to capture spatial and temporal dependencies simultaneously. Another work [22] learns the meta-knowledge from multiple cities and performs the spatial-temporal traffic prediction. However, these works still do not consider the temporal uncertainties in prediction and they do not extract meta-knowledge directly from traffic time series. Therefore, they have limited capabilities to learn traffic patterns that are rarely seen in the historical data.

In this paper, we try to solve the short-term traffic dynamics prediction problem and tackle the unique challenges mentioned before from the Bayesian meta-learning perspective. A novel continuous spatial-temporal meta-learner (cST-ML) 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 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 Sec III-B.)
- 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 Sec III-C and Sec III-D.)
- 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 Sec IV.) ***We have made our code and unique dataset available to contribute to the research community [1].***

TABLE I: 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 $\mathcal{T}_i$

The rest of the paper is organized as follows: Section II provides the overview of the paper, Section III details the design of cST-ML. We present the experimental results in Section IV and discuss related work in Section V. Finally, we conclude the paper in Section VI.

II. Overview

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

A. Problem definition

In a city, *urban traffic status* can be characterized by many statistics, such as traffic volumes, speed, inflow/outflow, *etc*, which are of great interests 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.*, $1 \times 1km$), denoted as $\mathcal{S} = \{s_{ij}\}$, where $1 \leq i \leq I, 1 \leq j \leq J$.

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, *etc*. 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.

Definition 4 (Traffic status). Traffic status indicates the quality of traffic, which can be measured by traffic inflow/outflow,

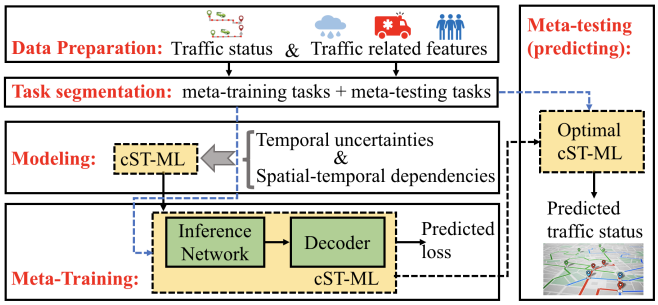


Fig. 3: Insight of the framework.

average driving speed, *etc*. We denote y^t as the average traffic status of grid cell s in time slot t.

In this paper, 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.

Definition 5 (Traffic prediction task.) A traffic prediction task $\mathcal{T}_i$ is composed of a set of paired $(\boldsymbol{X}^t, y^t)$ 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.*, $\mathcal{T}_i = \{(\boldsymbol{X}_i^1, y_i^1), \ldots, (\boldsymbol{X}_i^{N_t}, y_i^{N_t})\} = \{\mathcal{D}_i^{tr}, \mathcal{D}_i^{ts}\}$.

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 $\{\boldsymbol{X}^t\}$. 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 $p(\mathcal{T})$:

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

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:

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

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

B. 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 which 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.

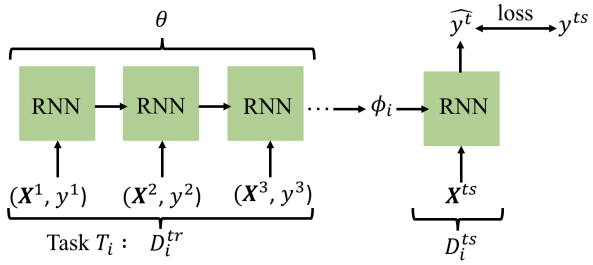


Fig. 4: Deterministic black-box meta-learning.

The detailed data preparation and task segmentation process will be presented in Section IV. We will first introduce the methodologies in the next section.

III. Methodologies

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

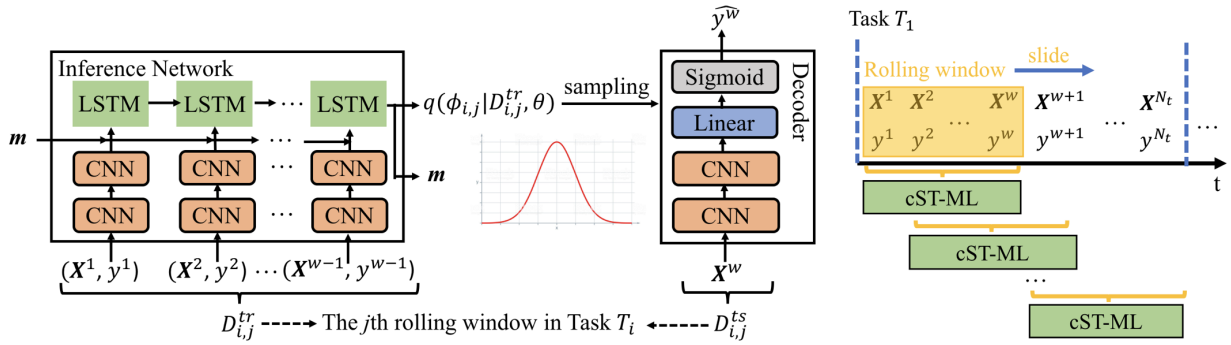
A. Key challenges

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 which are sample 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 Eq. 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 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 $q\left(\phi_i|\mathcal{D}_i^{\text{tr}}, \theta\right)$ is deterministic in this setup, which means there is no uncertainties in $\mathcal{D}_i^{\text{tr}}$, *i.e.*, $\phi_i = f_\theta\left(\mathcal{D}_i^{\text{tr}}\right)$. 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.

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

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



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

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

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. Since the adapted parameters ϕ_i is sampled, it can capture the uncertainties of tasks during each adaptation process.

Challenges. The Bayesian black-box meta-learning only captures the uncertainties between 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.

B. cST-ML Modeling

To address the challenges highlighted above, now we are in a position to develop continuous spatial-temporal meta-learning (cST-ML) framework.

Follow the original Bayesian black-box meta-learning, to deal with the uncertainties in task adaptation, we treat the adapted parameters as a latent variable. We approximate the likelihood with variational lower bound (ELBO), the ELBO is derived in Eq. 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 $\mathcal{D}_i^{\text{tr}}$ 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 $\mathcal{D}_{i,j}^{\text{ts}}$. Eq.5 is the log likelihood lower bound of the jth rolling window in task $\mathcal{T}_i$:

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

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 [11]). Compared with Eq.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 Eq.6.

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

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

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

C. 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 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 $\mathcal{D}_{i,j}^{\text{tr}}$ and extracts information from $\mathcal{D}_{i,j}^{\text{tr}}$, 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) $\mathcal{D}_{i,j}^{\text{tr}} = \{(\boldsymbol{X}_i^1, y_i^1), \ldots, (\boldsymbol{X}_i^t, y_i^t)\}$, 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.

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 $\mathcal{T}_i \sim p(\mathcal{T})$.
  Prepare $\mathcal{D}^{\text{tr}}_{i,j}$ and $\mathcal{D}^{\text{ts}}_{i,j}$ for each rolling window in $\mathcal{T}_i$
  for all rolling windows in $\mathcal{T}_i$ **do**
    Sample $\phi_{i,j}$ from $q\left(\phi_{i,j}|\mathcal{D}^{\text{tr}}_{i,j}, \theta\right)$.
    Compute log likelihood using Eq.5.
  end for
  Update θ with Adam [10] to maximize Eq.6.
end while

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 $\{\hat{y}^t, \ldots, \hat{y}^{N_t}\}$.

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 $\mathcal{T}$
for all rolling windows in $\mathcal{T}$ **do**
  $\hat{y}^t = f_{\theta^\star}(\mathcal{D}^{\text{tr}}, \mathcal{D}^{\text{ts}})$.
  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.
end for

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 $q\left(\phi_{i,j}|\mathcal{D}^{\text{tr}}_{i,j}, \theta\right)$.

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 $q\left(\phi_{i,j}|\mathcal{D}^{\text{tr}}_{i,j}, \theta\right)$ 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).

D. 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 $\{(\boldsymbol{X}^1_i, y^1_i), \ldots, (\boldsymbol{X}^{w-1}_i, y^{w-1}_i)\}$ in $\mathcal{T}_i$ as input of the inference network and use $\boldsymbol{X}^w_i$ as the input of decoder to predict $\hat{y}^w_i$, thus, $\mathcal{D}^{\text{tr}}_{i,j} = \{(\boldsymbol{X}^1_i, y^1_i), \ldots, (\boldsymbol{X}^{w-1}_i, y^{w-1}_i)\}$ and $\mathcal{D}^{\text{ts}}_{i,j} = \{(\boldsymbol{X}^w_i, y^w_i)\}$, just as illustrated in Figure 5. Then, we use data points $\{(\boldsymbol{X}^2_i, y^2_i), \ldots, (\boldsymbol{X}^w_i, y^w_i)\}$ 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 $\mathcal{D}^{\text{tr}}_{i,j}$ and $\mathcal{D}^{\text{ts}}_{i,j}$, 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.

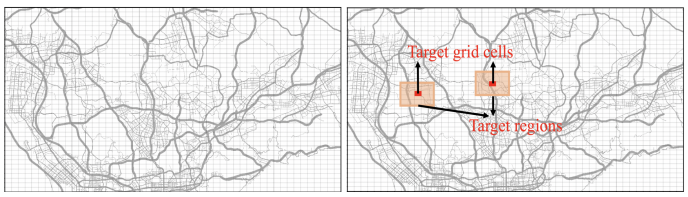
IV. Evaluation

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.

A. 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 Jul 1st to Dec 31st. 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.



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

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

Traffic speed, taxi inflow and travel demand are all extracted from taxi GPS records collected in Shenzhen, China from Jul 1st to Dec 31st, 2016. In each time slot (*i.e.*, one hour) of each day, taxi inflow is the number of taxis that stay or arrive at a

TABLE II: Performance on traffic speed prediction and taxi inflow prediction.

Methods			HA	Regression	ARIMA	LSTM	SNAIL	BBML	**cST-ML**
Traffic speed	1h	RMSE	2.993	2.224	2.160	2.923	2.216	6.805	**1.055**
		MAPE	0.170	0.124	0.124	0.156	0.126	0.379	**0.058**
	3h	RMSE	2.545	2.249	2.517	2.674	2.278	5.408	**2.119**
		MAPE	0.126	0.112	0.110	0.125	0.114	0.304	**0.093**
	6h	RMSE	3.120	3.035	3.711	3.000	2.933	5.360	**2.685**
		MAPE	0.197	0.183	0.221	0.199	0.171	0.328	**0.164**
Taxi inflow	1h	RMSE	56.833	37.356	26.902	37.501	29.715	24.749	**16.461**
		MAPE	0.239	0.159	0.120	0.160	0.116	0.104	**0.061**
	3h	RMSE	65.929	38.572	35.551	38.940	31.191	31.382	**19.258**
		MAPE	0.237	0.145	0.119	0.147	0.097	0.117	**0.064**
	6h	RMSE	64.777	31.937	38.734	32.781	25.592	29.253	**18.744**
		MAPE	0.235	0.111	0.124	0.114	0.079	0.106	**0.061**

target grid cell, travel demand is the number 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, and many studies have shown that taxi demand is a very representative measure of travel demand [8], [16].

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.

B. Baselines

- **HA [22]** For each grid cell, Historical Average method (HA) predicts the traffic status for a target grid cell based on its average status of the previous time slots.
- **Regression [2].** 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 [20].** 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.
- **LSTM [15], [27].** 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 [14].** It is an 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.
- **BBML [6].** It is the Bayesian black-box meta learning method without memory vector and rolling windows. The structure of this baseline is same as cST-ML, however, it does not apply rolling windows and there is no memories kept within a task.

C. Evaluation Metrics

We use mean absolute percentage error (MAPE) and rooted mean square error (RMSE) for evaluations:

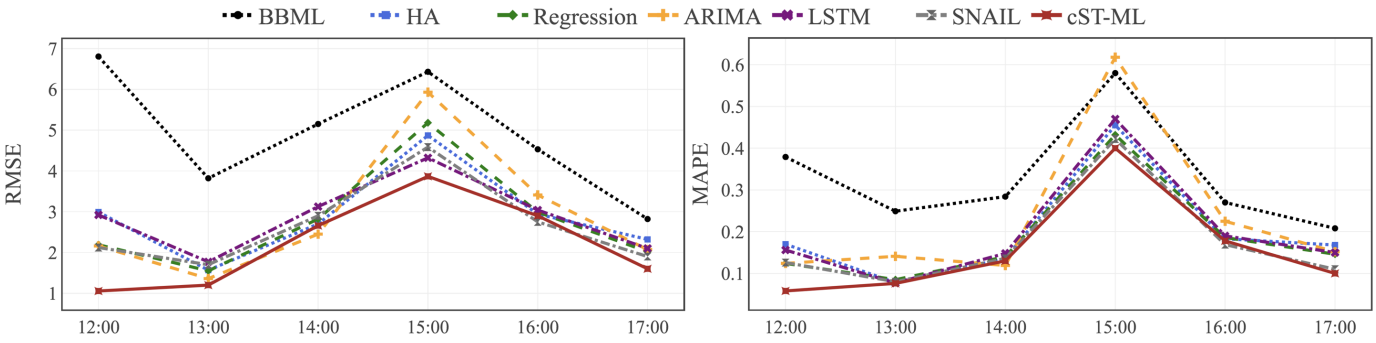
$$\text{MAPE} = \frac{1}{T}\sum_{t=1}^{T} |y_t - \hat{y}_t| / y_t, \tag{8}$$

$$\text{RMSE} = \sqrt{\frac{1}{T}\sum_{t=1}^{T} (y_t - \hat{y}_t)^2}, \tag{9}$$

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

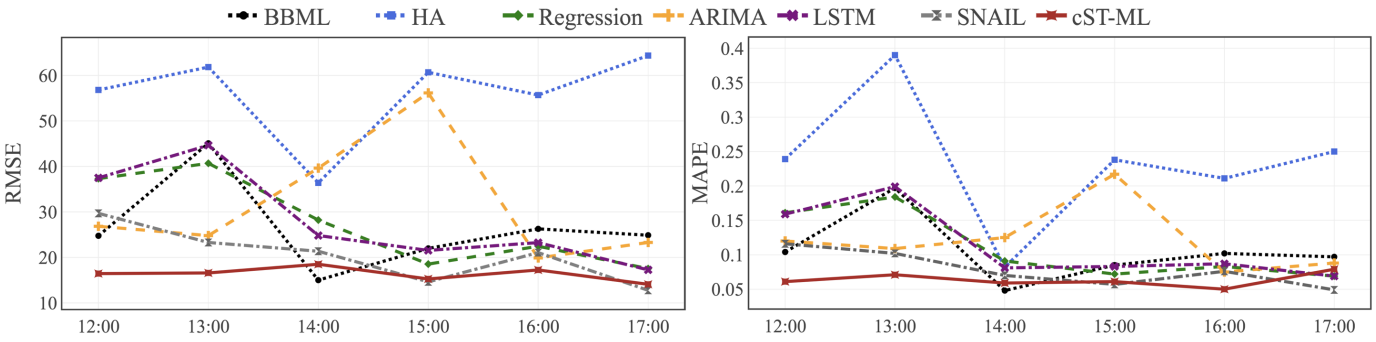
D. Experimental Settings

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.



(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.

The time interval for each task used to train the cST-ML are from 7:00am to 7:00pm, 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 [10] with $\beta_1 = 0.5$ and $\beta_2 = 0.999$, and a learning rate of 2×10^{-4} for 2000 times task samplings, the window size is 5 with step size equal to 1.

E. Evaluation Results

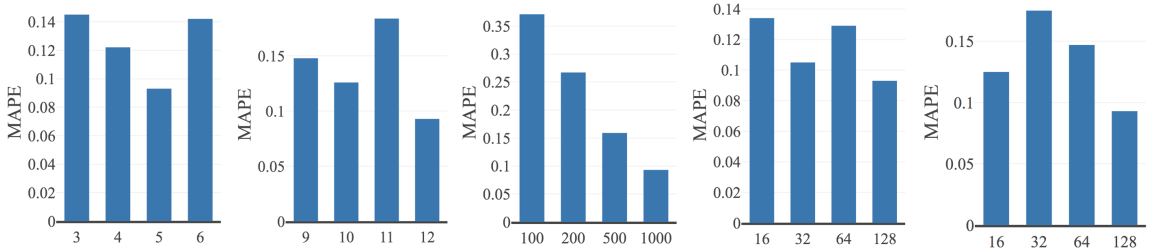
1) Average prediction performance: First, we conduct experiments to compare the average prediction performance of our proposed cST-ML and six competing baseline models. The results are shown in Table II. In the table, for a specific target grid cell, we present the RMSE and MAPE results for one-hour, three-hour and six-hour traffic speed prediction and taxi inflow prediction. For meta-based models (including cST-ML, BBML and SNAIL), we randomly pick 5 meta-testing tasks in both of the traffic speed and taxi inflow predictions, compute the one-hour, three-hour and six-hour 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 one-hour, three-hour and six-hour predictions, cST-ML outperforms all the baseline models. Compared with BBML model, cST-ML always has great improvements in both metrics, even though the two methods are both Bayesian black-box meta-learning based. The reason is that in 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 which lead to better prediction performance.

SNAIL is a deterministic black-box meta-learning method which does not consider any uncertainties of tasks, but it achieves competitive performance in some cases (*i.e.*, six-hour 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.

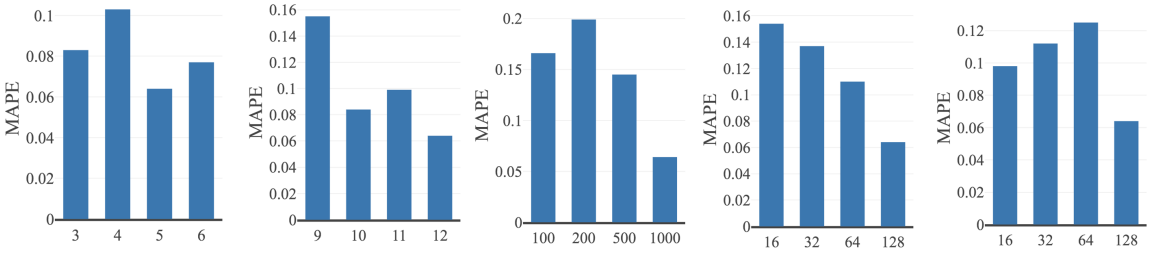
LSTM is used as a seq2seq model in traffic prediction, which utilizes the traffic related features to predict the traffic status, so it does not rely on the previous traffic status in testing or prediction process which could result in larger prediction errors.

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



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

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



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

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

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 and BBML are 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.

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.*, one hour). 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.*, one hour). The statistics are calculated based on 5 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 5 testing tasks.

In traffic speed prediction, the detailed performance is presented in Figure 7. As shown in Figure 7(a) and Figure 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 Figure 8(a) and Figure 8(b), our cST-ML also achieves the best prediction performance except in a few time slots, for example, BBML and SNAIL have 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 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.

3) Evaluations on cST-ML parameters: cST-ML has many hyper-parameters, *e.g.*, rolling window size, task length, *etc*. In this part, we conduct experiments to evaluate the impacts of different hyper-parameters on our cST-ML. In Figure 9 and Figure 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 Figure 9 and Figure 10 are MAPEs of 3-hour predictions, which are computed using 5 meta-testing tasks in both of traffic speed prediction and taxi inflow prediction.

As shown in Figure 9(a) and Figure 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.*, 2 or 3 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 Figure 9(b) and Figure 10(b), task length also influences

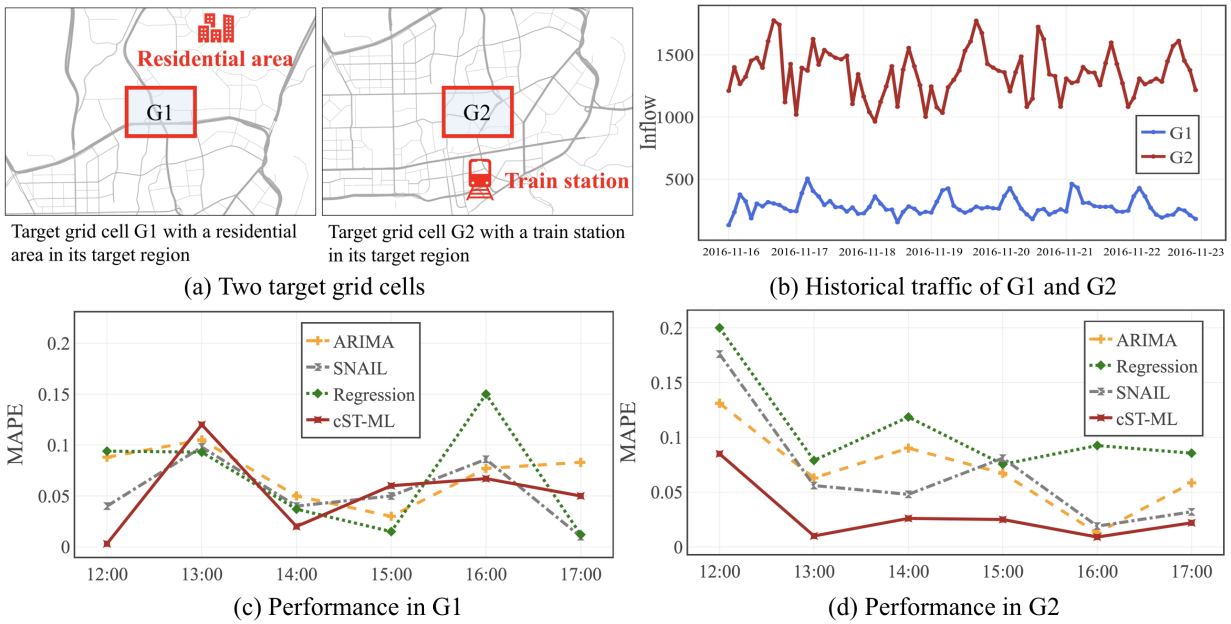


Fig. 11: Traffic predictions of two target grid cells.

the performance of our model and when task length is equal to 12, we have the best performance. Since the historical traffic data (from 7:00am to 7:00pm every day) can be viewed as time series data, if we segment the tasks by 12 hours, 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 hours, 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 Figure 9(c) and Figure 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.

We also change the dimensions of hidden states in LSTM (inside the inference network of cST-ML). The results are shown in Figure 9(d) and Figure 10(d). We find that the model performance is very sensitive to the dimension of hidden states, higher dimension leads to better performance, which indicates more spatial and temporal information is kept by LSTM.

Similarly, we analyze how the dimension of mean and log variance influence performance in Figure 9(e) and Figure 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.

F. 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 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 Figure 11(c) and (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.

V. Related Work

Urban Traffic Prediction. In urban traffic prediction area, some works focused on traffic volume and crowd flow prediction. For example, one work [25] proposed a citywide traffic volume estimation framework which combined machine learning techniques and traffic flow theory. Another work [21] 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 [3], [19] and [4] applied support vector regression to predict future traffic and took environmental features into consideration. The paper [12] proposed a traffic prediction method which combined SARIMA model and autoregressive model with genetic algorithm optimization. Another paper [26] tried to predict citywide flow using CNN which better captured traffic spatial dependencies. Yuan et al. [24] tried to predict traffic accidents with ConvLSTM. 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 [7], Reptile [17], SNAIL [14], MOCA [9], *etc*. MAMAL 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, the work [18] combined meta graph attention and meta recurrent neural network to capture spatial and temporal dependencies simultaneously. Another work [22] learn the meta-knowledge from multiple cities and perform the spatial-temporal traffic prediction. However, these two works still did not consider the task and temporal uncertainties in prediction.

VI. Conclusion

In this paper, 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] cST-ML. https://github.com/cST-ML/cST-ML, 2020. [Online].

[2] I. Alam, D. M. Farid, and R. J. F. Rossetti. The prediction of traffic flow with regression analysis. In *IEMIS*, pages 661–671, 2019.

[3] M. Castro-Neto, Y.-S. Jeong, M.-K. Jeong, and L. Han. Online-svr for short-term traffic flow prediction under typical and atypical traffic conditions. *Expert Systems with Applications*, 36:6164–6173, 2009.

[4] Y. Cong, J. Wang, and X. Li. Traffic flow forecasting by a least squares support vector machine with a fruit fly optimization algorithm. *Procedia Engineering*, 137:59 – 68, 2016.

[5] Z. Cui, R. Ke, and Y. Wang. Deep stacked bidirectional and unidirectional lstm recurrent neural network for network-wide traffic speed prediction. In *6th International Workshop on Urban Computing*, 2017.

[6] C. Finn. Bayesian meta-learning. https://cs330.stanford.edu/slides/cs330_bayesian_metalearning.pdf, 2019. [Online].

[7] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, page 1126–1135, 2017.

[8] E. J. Gonzales, C. J. Yang, E. F. Morgul, and K. Ozbay. Modeling taxi demand with gps data from taxis and transit. Technical report, Mineta National Transit Research Consortium, 2014.

[9] J. Harrison, A. Sharma, C. Finn, and M. Pavone. Continuous meta-learning without tasks, 2019.

[10] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[11] D. P. Kingma and M. Welling. Auto-encoding variational bayes, 2013.

[12] X. Luo, L. Niu, and S. Zhang. An algorithm for traffic flow prediction based on improved sarima and ga. *KSCE Journal of Civil Engineering*, 22:1–9, 05 2018.

[13] Y. Lv, Y. Duan, W. Kang, Z. Li, F.-Y. Wang, et al. Traffic flow prediction with big data: A deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):865 – 873, 2015.

[14] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel. A simple neural attentive meta-learner. In *International Conference on Learning Representations*, 2018.

[15] O. Mogren. C-RNN-GAN: continuous recurrent neural networks with adversarial training. *CoRR*, 2016.

[16] N. Mukai and N. Yoden. Taxi demand forecasting based on taxi probe data by neural network. In *IIMSS*, pages 589–597, 2012.

[17] A. Nichol, J. Achiam, and J. Schulman. On first-order meta-learning algorithms. *CoRR*, 2018.

[18] Z. Pan, Y. Liang, W. Wang, Y. Yu, Y. Zheng, and J. Zhang. Urban traffic prediction from spatio-temporal data using deep meta learning. In *KDD*, 05 2019.

[19] Y. Sun, B. Leng, and G. Wei. A novel wavelet-svm short-time passenger flow prediction in beijing subway system. *Neurocomputing*, 166, 04 2015.

[20] M. Tan, S. C. Wong, J. Xu, Z. Guan, and P. Zhang. An aggregation approach to short-term traffic flow prediction. *IEEE Transactions on Intelligent Transportation Systems*, pages 60–69, 2009.

[21] E. Toto, E. A. Rundensteiner, Y. Li, R. Jordan, M. Ishutkina, K. Claypool, J. Luo, and F. Zhang. Pulse: A real time system for crowd flow prediction at metropolitan subway stations. In *ECMLPKDD*, 2016.

[22] H. Yao, Y. Liu, Y. Wei, X. Tang, and Z. Li. Learning from multiple cities: A meta-learning approach for spatial-temporal prediction. In *The World Wide Web Conference*, page 2181–2191, 2019.

[23] H. Yu, Z. Wu, S. Wang, Y. Wang, and X. Ma. Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks. *Sensors*, 17(7), 2017.

[24] Z. Yuan, X. Zhou, and T. Yang. Hetero-ConvLSTM: A deep learning approach to traffic accident prediction on heterogeneous spatio-temporal data. In *KDD*, pages 984–992, 2018.

[25] X. Zhan, Y. Zheng, X. Yi, and S. Ukkusuri. Citywide traffic volume estimation using trajectory data. *TKDE*, 29(2):272–285, 2017.

[26] J. Zhang, Y. Zheng, and D. Qi. Deep spatio-temporal residual networks for citywide crowd flows prediction. *CoRR*, 2016.

[27] J. Zhao, F. Deng, Y. Cai, and J. Chen. Long short-term memory - fully connected (lstm-fc) neural network for pm2.5 concentration prediction. *Chemosphere*, 2019.

[28] Y. Zheng, L. Capra, O. Wolfson, and H. Yang. Urban computing: concepts, methodologies, and applications. *TIST*, 2014.