Edge Computing Resources Reservation in Vehicular Networks: A Meta-Learning Approach

Dawei Chen, *Student Member, IEEE*, Yin-Chen Liu, BaekGyu Kim, *Member, IEEE*, Jiang Xie, *Fellow, IEEE*, Choong Seon Hong, *Senior Member, IEEE*, and Zhu Han, *Fellow, IEEE*

***Abstract*—With the development of autonomous vehicular technologies, the execution tasks become more memory-consuming and computation-intensive. Simultaneously, a certain portion of tasks are latency-sensitive, such as collaborative perception, path planning, collaborative simultaneous localization and mapping, real-time pedestrian detection, etc. Because of the limited computation resources inside vehicles and restricted transmission bandwidth, edge computing can be an effective way to assist with the tasks execution. Considering from the perspective of business, the reservation or subscription cost is cheaper than real time requests. In order to minimize the expense of consuming edge services, the desirable situation is to reserve the resources as much as needed. However, the configuration of vehicular network is variational in practice due to the diversity of road maps, different time range like peak time and off-peak time, and the various task types, which makes it challenging to figure out a general machine learning model that is suitable for any case. Therefore, to predict the resource consumption in edge nodes accurately in different scenarios, we propose a two-stage meta-learning based approach to adaptively choose the appropriate machine learning algorithms based on the meta-features extracted on database. Besides, due to the deficiency of dataset for edge resource consumption, we program in game engine unity to generate the 3D model of Manhattan area. Meanwhile, we change the factors like different road maps and number of vehicles so as to get closer to practices. In the evaluation part, we adopt root mean square error, mean absolute percentage, and mean GEH as evaluation metrics to assess the performance of each model. Also, a quantitative analysis for the total cost and waste is also conducted. Eventually, we can find that the proposed meta-learning based method outperforms the non-meta ones.**

Manuscript received December 11, 2019; revised February 14, 2020; accepted March 19, 2020. Date of publication March 31, 2020; date of current version May 14, 2020. This work was supported in part by US MURI AFOSR MURI 18RT0073, in part by NSF EARS-1839818, in part by CNS1717454, in part by CNS-1731424, in part by CNS-1702850, in part by US National Science Foundation (NSF) under Grants 1718666, 1731675, 1910667, and 1910891, and in part by Toyota Motor North America. The review of this article was coordinated by Dr. B. Mao. *(Corresponding author: Choong Seon Hong.)*

Dawei Chen is with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77004 USA (e-mail: dchen22@uh.edu).

Yin-Chen Liu and BaekGyu Kim are with the Toyota Motor North America, Inc., Mountain View, CA 94043 USA (e-mail: yin-chen.liu@toyota.com; baekgyu.kim@toyota.com).

Jiang Xie is with the Department of Electrical and Computer Engineering, University of North Carolina at Charlotte, Charlotte, NC 28223 USA (e-mail: jxie1@uncc.edu).

Choong Seon Hong is with the Department of Computer Science and Engineering, Kyung Hee University, Yongin-si, Gyeonggi-do 17104, South Korea (e-mail: cshong@khu.ac.kr).

Zhu Han is with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77004 USA, and also with the Department of Computer Science and Engineering, Kyung Hee University, Seoul 446-701, South Korea (e-mail: zhan2@uh.edu).

Digital Object Identifier 10.1109/TVT.2020.2983445

***Index Terms*—Edge computing, resource reservation, meta-learning, vehicular network.**

I. Introduction

HAVING stepped into the era of information technology, there are enormous artificial intelligence based autonomous devices, technologies, and services coming into being, one important branch of which is autonomous vehicles or intelligent vehicles. According to the definition of National Highway Traffic Safety Administration (NHTSA), the levels of vehicle automation can be categorized into six classes, which are distinguished by the extent of autonomy [1]. Currently, the performance of autonomous vehicles can just meet the requirements between level 2 and level 3, and both of which require the driver must be ready to take back control at any time. In other words, the artificial intelligence based autonomous driving remains much to be done before realizing the human occupants never need to be involved in driving, such as accurate prediction, precise inference, latency decreasing, etc.

For the time being, the artificial intelligence technologies of autonomous vehicles heavily rely on the data generated by the built-in devices such as an array of sensors, electronic control units, cameras, etc. According to the forecast of Intel, the data generated by one single autonomous vehicle will achieve 4 TB data per day [2]. Such a kind of massive data bring inevitable challenges to data processing and storage within vehicles, especially for those real-time tasks with high computation complexity such as collaborative perception, path planning, collaborative simultaneous localization and mapping (SLAM), real-time pedestrian detection, or with high demands for storage capacity like uploading driving records. In this case, cloud computing can be an effective way to help.

Nowadays, there are two mainstream cloud computing paradigms: one is the conventional centralized cloud computing and the other one is edge computing. The superiority of traditional centralized cloud computing is founded on the powerful data processing ability and the enormous storage capacity of remote datacenter. However, as a centralized paradigm, all the data needs to be transmitted to the datacenter for storage or further processing. The latency caused by long transmission distance and limited bandwidth is a significant challenge for vehicular networks [3]–[6]. On the opposite side, the architecture of edge computing performs better for latency alleviation and is more suitable for real-time tasks demanded by autonomous vehicles. Because in edge computing configuration, many edge

nodes will be deployed in a geographically distributed manner within the network, which means the services can be provided to the users more closely. Moreover, edge nodes are capable of computation ability and storage capacity to a certain extent, which makes it feasible to perform tasks locally and feedback the results to vehicles on time. Accordingly, transmission latency can be reduced significantly and quality of service (QoS) can be remarkably improved [7]–[10].

From the commercial perspective, the investment for edge nodes deployment will increase the expenditures of services-providers correspondingly, resulting in a relatively expensive price of edge services [11]. Meanwhile, aiming at catering to the market demands and attracting more customers, the corporations carry out some marketing strategies, one of which is that the services can be sold in a reservation or subscription way with a cheaper price and a real-time requested way with an expensive price. Furthermore, different purchasing programs are supplied and the customers can decide which program to get enrolled in according to their own consumption characteristics. For instance, Amazon Web Services (AWS) provides several purchase schemes to customers, which are pay-as-you-go, save when you reserve, and pay less by using more [12]. Concretely, these programs are described as the following:

1) Pay-as-you-go: the customers can adjust the services demands at any time depending on the their own needs and only need to pay for services on an as needed basis.
2) Save when you reserve: for the service like the Amazon Elastic Compute Cloud (Amazon EC2) and Amazon Relational Database Service (Amazon RDS), if the customers reserve in advance, a certain percent discount will be provided. In addition, if paid with upfront payments, the customers will be charged further less.
3) Pay less by using more: this is a common marketing strategy, which means for services like Amazon simple storage services (S3), if more volumes you consume, the less you will pay per GB.

Obviously, it is a more economical way to consume services by reservation in advance and the customers can save up to 75% compared with equivalent on-demand scheme based on the AWS's description. Therefore, for the purpose of minimizing the expenditure, it is of great importance for customers to figure out how many resources, i.e., computational memory or storage memory, should be reserved. Intuitively, this expense minimization problem can be regarded as a prediction problem. However, the amount of edge resource consumption actually is closely related to many factors, such as speed, road map, task memory consumption, etc. While traditional optimization algorithms are not effective to address such high-dimensional nonlinear regression problems [13]. Fortunately, the emerging machine learning methods provide powerful tools to tackle with such prediction problems. Due to the diversity of influence factors in vehicular networks, we can hardly find one machine learning model which is suitable and performs the best in any scenarios [14]. Considering there are similarities in these scenarios, meta-learning can be an effective method to help, whose goal is to learn from the experience or prior knowledge that generalizes well to related new tasks. Therefore, we propose a two-stage meta-learning based approach to adaptively choose the appropriate machine learning algorithms according to the meta-features extracted on databases.

In details, the contributions of this paper can be summarized as follows:

- The majority of existing edge computing-related papers focus on the perspective of edge services provider or the technological aspect of edge computing networks, such as minimizing energy consumption, maximizing QoS, minimizing transmission delay, etc. In this work, we consider the business perspective from edge services consumers and aim at minimizing the customers' cost for consuming edge node resources, which is seldom done in existing literature.
- Because of the diversity of factors in vehicular networks, such as time (like peak or off-peak), road map, and task type, we propose a two-stage meta-learning based method to adaptively select the appropriate machine learning model, which anticipates accurately and correspondingly gives the lowest expenditure in different scenarios.
- Since there is no open dataset about edge resource consumption, we program on the game engine unity to build the 3D model of Manhattan area to generate the data. At the same time, we change the factors to test our method in various scenarios, including road maps, number of vehicles (to mimic the peak and off-peak time), and the randomness of memory consumption size, so that the experiment environment is able to get closer to the practice.

The structure of the rest paper is as following. Section II discusses some related existing papers from both engineering field and business field. Section III introduces the specific scenario and the problem needs to be solved. Section IV introduces the proposed two-stage meta-learning approach and the machine learning models implemented in this work. Section V firstly introduces the process of data generation in detail and shows the performances of our proposed method comparing with the non-meta methods. Finally, a conclusion is drawn in Section VI.

II. Related Work

As a promising distributed computation paradigm, edge computing has become a popular field of research. However, most existing papers in an edge computing scenario focus on the technology perspective of the edge computing network side, like network architecture improvements, edge nodes placement, edge-assisted tasks offloading, etc. [15] proposed a fine-grained collaborative offloading strategy with caching enhancement scheme to minimize the latency at the edge side in both femto-cloud mobile network scenario and mobile edge computing scenario. [16] proposed a method based on the Lagrangian heuristic algorithm and workload allocation scheme to optimally place the cloudlets, under the considerations of both cloudlet cost and average end-to-end delay in a mobile edge computing scenario. [17] proposed a novel communication scheme to enable the low-latency, robust and accurate edge node assisted self-driving service for connected autonomous driving services

TABLE I
TYPICAL EDGE COMPUTING COMPANIES AND PRODUCTS

Company Name	Typical Product	Main Functionalities
Amazon	AWS Greengrass	Pushing local computing, communication, caching, sync, and machine learning inference to edge devices
Clear Blade	Edge Software	Deploying and managing IoT systems on the edge, and communicating with on-premise devices
Cisco	Gateway IC3000	Providing built-in security and manageability, enabling faster decision making at the edge
Dell EMC	VMware Cloud	Extending public cloud benefits to workloads in both private datacenter and edge locations
FogHorn	Lightning	Offering rapid data ingestion, sensor fusion, and machine learning to generate actionable insights in real-time
HPE	Edgeline	Aggregating and filtering data, analyzing video streams, and translating industrial protocols
IBM	Waston IoT	IoT security, data analyzing, IoT management, and IoT machine learning application
Microsoft	Azure IoT Edge	Offloading artificial intelligence and analytics workloads to the edge
Rigado	Cascade	Edge infrastructure establishment and connecting devices in Commercial IoT environments
Saguna	vEdge	Providing virtualized resources to enable on-edge applications and real-time network visibility

in a mobile edge computing scenario. Different from these literature, this paper stands on the side of edge resources consumers and focuses on the economic aspect to minimize the expenditure for customers. Therefore, in this paper, the specific architecture of edge computing network, the limitation of edge resources, and the wireless communications are unconcerned.

Thanks to the low-latency and distributed properties of edge computing, diverse use cases can be supported, such as connected autonomous vehicles, e-health, industrial automation, mobile gaming, smart grid, and Internet of Things (IoT) services [18]–[21]. Correspondingly, such characteristics and capabilities offered by an edge computing platform can be translated into unique value and revenue generation [22], which also prompts some researches on commercial aspect of edge computing. Present typical edge computing companies are summarized in Table I [23] discussed the influences of five factors, i.e., ease of use, security, cost reduction, reliability, and collaborating, on the cloud usages of micro and small businesses. [24] introduced a framework that leverages pricing aspects to enable the sharing economy vision for edge devices applied into the smart city scenario. [25] compared several pricing models, such as the time based model, volume based model, flat rate, content based model, etc., and discusses the pricing schemes from different cloud services providers including AWS EC2, AWS S3, Microsoft Azure, and AppNexus. Whereas, these works analyze and discuss problems on from the perspectives of service providers and the market operations instead of service customers as well.

As a promising machine learning approach, meta-learning has attracted considerable interests of diverse science and engineering communities recently, and is widely used in different fields. [26] proposed a meta-learning based framework to learn the online learning algorithm from offline videos so as to address an object tracking problem. [27] proposed a meta-learning based method to tackle with an automatic text classification problem through utilizing the distance-based meta-features derived from the original bag-of-words representation. [28] proposed a novel meta-learning method for domain generalization by a model agnostic training procedure so that the domain shift problem can be avoided. [29] proposed a two deep neural network architecture based meta-learning strategy to solve the clod-start problem for item recommendations when new items arrive continuously. However, to the best of our knowledge, there is no existing literature that implements a meta-learning based method to deal with an edge resource reservation problem in vehicular networks.

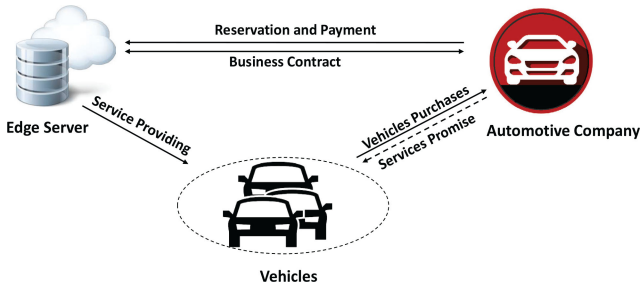


Fig. 1. The scenario and procedures description.

III. SCENARIO DESCRIPTION AND PROBLEM FORMULATION

In this paper, we consider an edge computing platform deploying edge nodes and providing edge computing services. The computational resources or storage resources can be purchased via reservation or real-time request. The automotive company provides services to the autonomous vehicle customers, such as collaborative perception, path planning, collaborative simultaneous localization and mapping, real-time object detection, etc. Consequently, it is inevitable for automotive company to pay the rental fee or operation fee to the edge platform so as to offer the corresponding services to customers. The procedures are described in Fig. 1.

We define the unit price for reservation and real-time request as P_{rv} and P_{rt}, respectively, and generally $P_{rv} \leq P_{rt}$. To perform some computation-intensive or storage-demanding tasks, at each time t the practical needed amount for edge resources is $n^{(t)}$ units. While the total amount reserved by automotive company is $m^{(t)}$ units. Intuitively, only when $n^{(t)} = m^{(t)}$, the consumption expenditure is optimally minimized. Because when $n^{(t)} > m^{(t)}$, the reserved amount of resources cannot meet the usage requirements and the extra real-time purchasing is inevitable. When $n^{(t)} < m^{(t)}$, the reserved amount of resources exceeds the actual demand, which actually is a waste for both customers and services provider. Therefore, the total cost of consuming edge services $S^{(t)}$ can be described by the following

function:

$$S(t) = \begin{cases} m^{(t)} \times P_{\text{rv}}, & n^{(t)} \leq m^{(t)}, \\ m^{(t)} \times P_{\text{rv}} + (n^{(t)} - m^{(t)}) \times P_{\text{rt}}, & m^{(t)} < n^{(t)}. \end{cases} \tag{1}$$

Correspondingly, the total waste $W^{(t)}$ can be calculated as the following piece-wise function:

$$W^{(t)} = \begin{cases} (m^{(t)} - n^{(t)}) \times P_{\text{rv}}, & n^{(t)} \leq m^{(t)}, \\ (n^{(t)} - m^{(t)}) \times (P_{\text{rt}} - P_{\text{rv}}), & m^{(t)} < n^{(t)}. \end{cases} \tag{2}$$

Obviously, in order to minimize the waste, the desirable situation is $m^{(t)} = n^{(t)}$. Thereby, the objective function can be defined by mean-square-error (MSE):

$$\min \left\| m^{(t)} - n^{(t)} \right\|_2^2. \tag{3}$$

IV. Methodology

In this section, we firstly introduce our proposed two-stage meta-learning approach. Then, the machine learning models utilized in this paper are introduced as well.

A. Meta-Learning

Meta-learning is also known as learning to learn, which is not specifically defined but covers any kind of machine learning methods based on prior knowledge learned from other related tasks. One of the typical meta-learning problems is the algorithm selection problem (ASP). [30] formulates the classical ASP and proves that there is a connection between the problem characteristics and the algorithm which can be utilized to solve it. Later, [31] further demonstrates there will be no one single algorithm that can work out optimums for all the problems. In other words, the algorithm performs differently from problem to problem, or more specifically, from dataset to dataset. However, generally, ASPs are NP-hard, which are challenging to be resolved by traditional optimization methods [32]. Therefore, we propose a two-stage meta-learning approach to address this problem. Before looking into the details, some preliminary definitions and descriptions are introduced here. In this work, the meta-learning task is made up by the following components:

- The task or problem space $\mathcal{P}$: a set of problems that contains both solved and unsolved ones, which specifically can be a regression or classification problem in supervised learning or a state-space action decision problem in reinforcement learning. Essentially, the dimension of $\mathcal{P}$ is high due to the diversity of characteristics embedded in the tasks, which is the factor that leads to the performance fluctuation of same model among different problems. In this work, $\mathcal{P}$ indicates the edge resource prediction problems in vehicular networks under different scenarios, which can be considered as regression problems substantially.
- The meta-feature space $\mathcal{C}$: a set of data characteristics reflecting the internal or intrinsic representations of a dataset. Basically, $\mathcal{C}$ is a multi-dimensional vector mathematically. The specific types of meta-features are decided by the problem and the selected machine learning models, the typical forms of which include simple meta-features (such as the number of samples, the number of classes, etc.), statistical meta-features (such as skewness, kuitosis, covariance, correlation, etc.), information-theoretic meta-features (such as norm entropy, mutual information, uncertainty coefficient, etc.), complexity based meta-features (such as Fisher's discriminant, volume of overlap, data consistency, etc.), model based meta-features (like for decision tree, the number of leaves, branch length, information can be the meta-features), and landmarkers, etc [33]. The specific meta-features utilized in this work are introduced in details in Section V-A.
- The machine learning model or algorithm space $\mathcal{M}$: basically it can be the universal set which contains all the existing algorithms. Whereas, practically, when considering one specific problem, $\mathcal{M}$ can only be a set of appropriately selected ones. In this work, because the tasks are time series data based prediction problems, $\mathcal{M}$ is defined as a group of long short term memory (LSTM) based machine learning models, which is detailedly introduced in Section IV-B.
- The performance evaluation space $\mathcal{E}$: is a set of diverse metrics to assess the performance of algorithms in $\mathcal{M}$ on a dataset. The evaluation metrics implemented in this work are discussed concretely in Section V-B.

For each task $\mathcal{T}_i$ in $\mathcal{P}$, what is supposed to be learned the distribution over the dataset, which can be described as $p(\mathcal{T}_i)$. In order to find the best regression model, the traditional method is to apply all the algorithms in $\mathcal{M}$ to $\mathcal{T}_i$, and then perform ranking calculations to determine which specific algorithm should be the one [34]. However, one defect of this approach is that, each time when dealing with a new but related task, calculating over the space $\mathcal{M}$ is time-costing. Meanwhile, the prior-knowledge concealed in previous experiences are not well exploitative. Therefore, we propose a two-stage meta-learning method to solve this problem, which is illustrated in Fig. 2. On the first stage, we utilize a deep neural network (named as Decider) to figure out which algorithm should be selected according to the experiences. On the second stage, the chosen machine learning model (named as Prognosticator) is implemented to perform the inference for edge resource consumption.

Basically, each algorithm in $\mathcal{M}$ can be denoted as f_{θ_i}, where f is the function that can represent the i^{th} algorithm in $\mathcal{M}$ and θ describes the corresponding parameters determined by the machine learning model configurations. Intuitively, the objective function or loss function of the Decider can be written as

$$\begin{aligned} &\min_i \ \mathcal{L}_1\left(f_{\theta_i}\right) = \left\| f_{\theta_i}(\boldsymbol{c}) - r \right\|_2^2, \\ &\text{s.t. } f_{\theta_i} \in \mathcal{M}, \\ &\quad \boldsymbol{c} \in \mathcal{C}, \end{aligned} \tag{4}$$

where $\boldsymbol{c}$ is the meta-feature vector and r is the suggested algorithm or the best performance one according to previous experiences which gives the most economical scheme correspondingly. The architecture of the Decider is a fully-connected deep neural network, which means the output of each hidden layer will be the input of the following hidden layer. Thus, to optimize the

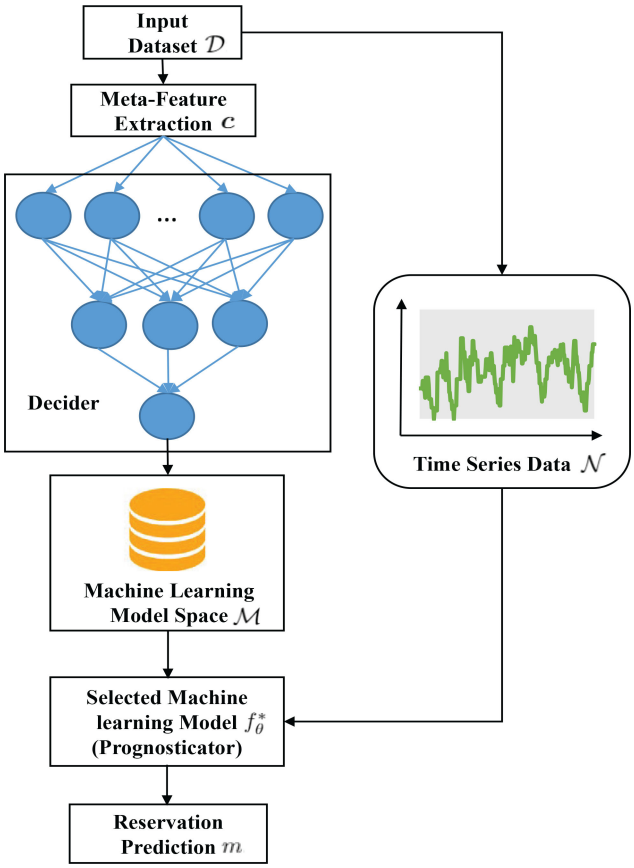


Fig. 2. The proposed meta-learning framework.

network, back-propagation is inevitable operation through the network, which can be described as,

$$\frac{\partial \mathcal{L}_1}{\partial \boldsymbol{\delta}^{(l)}} = \sum_l \left(\frac{\partial \mathcal{L}_1}{\partial o^{(l)}} \right) \frac{\partial o^{(l)}}{\partial o^{(l-1)}} \frac{\partial o^{(l-1)}}{\boldsymbol{\delta}^{(l)}}, \tag{5}$$

where l denotes the number of hidden layers of the Decider, $\boldsymbol{\delta}^{(l)}$ describes the generalized parameter set (including weight and bias) in layer l, i.e., $\boldsymbol{\delta}^l = \{\boldsymbol{w}^{(l)}, \boldsymbol{b}^{(l)}\}$, and $o^{(l)}$ denotes the output of layer l. During each iteration, the parameters of all the layers will get updated by

$$\boldsymbol{\delta}^{'} = \boldsymbol{\delta} - \alpha \nabla_{\boldsymbol{\delta}} \mathcal{L}_1, \tag{6}$$

where α is the learning rate. When the number of iterations or the value of loss function achieves the threshold, the training phase ends and the optimal algorithm f_θ^* or the Prognosticator can be obtained. Until here, all the operations in stage one have been finished and the corresponding procedures are summarized in Algorithm 1.

The second stage is to exploit the Prognosticator obtained in stage one to predict the edge resource consumption. In this stage, what needs to be done is to optimize the parameters of Prognosticator so as to fit the new dataset $\mathcal{N}$ from $\mathcal{T}_i$. Pre-requisitely, it is necessary to divide $\mathcal{N}$ into training set and testing set, which are denoted as $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively. Therefore, the objective

Algorithm 1: The First Stage Meta-Learning.

1: Input: experience data $\mathcal{D}$; meta-feature space $\mathcal{C}$; machine learning model space $\mathcal{M}$; new task $\mathcal{T}_i$; learning rate α; maximum iteration steps $t_{\max}$.
2: Randomly initialize the parameter $\boldsymbol{\delta}$.
3: **for** t = 1: $t_{\max}$ **do**
4: Feed forward propagate all the samples in $\mathcal{D}$ and calculate the MSE by (4);
5: Back propagate MSE through the network by applying (5) and update the values of parameter set $\boldsymbol{\delta}^{'}$ via (6);
6: **end for**
7: Output: the trained Decider with optimal parameters $\boldsymbol{\delta}^*$;

Algorithm 2: The Second Stage Meta-Learning.

1: Input: meta-feature space $\mathcal{C}$; machine learning model space $\mathcal{M}$; new task $\mathcal{T}_i$; new task dataset $\mathcal{N}$; learning rate β; maximum iteration steps $k_{\max}$.
2: Feed the meta-features $\boldsymbol{c}_i$ from $\mathcal{T}_i$ into the Decider and the specific Prognosticator, i.e., f_θ^*, can be obtained;
3: Divide $\mathcal{N}$ into training set $\boldsymbol{x}$ and testing set $\boldsymbol{y}$;
4: Feed $\boldsymbol{x}$ into f_θ^* with randomly initialized parameters $\boldsymbol{\phi}$;
5: **for** k = 1: $k_{\max}$ **do**
6: Feed forward propagate all the samples through $\boldsymbol{x}$ and get the MSE by (7);
7: Back propagate MSE throughout the network f_θ^* by applying (8) and update the parameter set $\boldsymbol{\phi}^{'}$ via (9);
8: **end for**
9: Output: the Prognosticator f_θ^* with optimal parameters $\boldsymbol{\phi}^*$;
10: Feed the testing data $\boldsymbol{y}$ into $f_\theta^*(\boldsymbol{\phi}^*)$ and infer the amount of edge resource consumption;

function of Prognosticator can be defined as

$$\min_{\boldsymbol{\phi}} \ \mathcal{L}_2\left(f_\theta^*\left(\boldsymbol{\phi}\right)\right) = \sum_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{T}_i} \left\| f_\theta^*\left(\boldsymbol{\phi};\boldsymbol{x}\right) - \boldsymbol{y} \right\|_2^2, \tag{7}$$

where $\boldsymbol{\phi}$ is the parameters set for the machine learning model f_θ^*. To minimize the error, likewise, the Prognosticator will perform the back-propagation, i.e.,

$$\frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\phi}^{(p)}} = \sum_p \left(\frac{\partial \mathcal{L}_2}{\partial o^{(p)}} \right) \frac{\partial o^{(p)}}{\partial o^{(p-1)}} \frac{\partial o^{(p-1)}}{\boldsymbol{\phi}^{(p)}}, \tag{8}$$

where p denotes the number of hidden layers in the Prognosticator. For step in (8), the parameters $\boldsymbol{\phi}$ will be updated by

$$\boldsymbol{\phi}^{'} = \boldsymbol{\phi} - \beta \nabla_{\boldsymbol{\phi}} \mathcal{L}_2, \tag{9}$$

where β is the learning step size. Overall, the second-stage procedures are summarized in Algorithm 2.

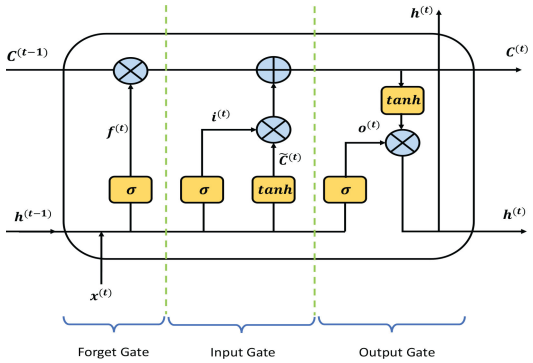


Fig. 3. The structure of a LSTM cell.

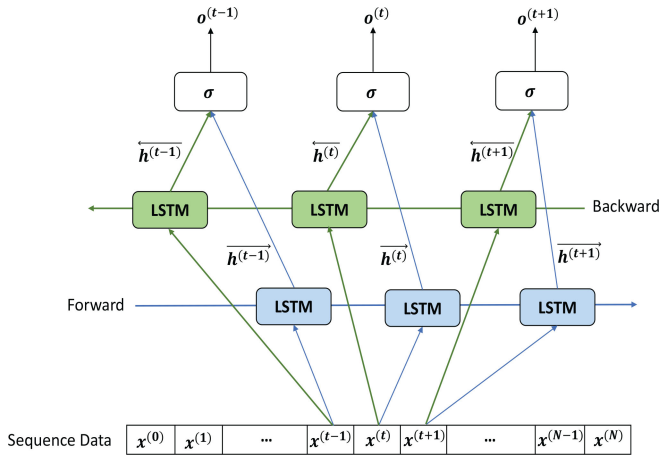


Fig. 4. The architecture of BiLSTM network.

B. Machine Learning Model Space

In this section, the algorithm space $\mathcal{M}$ are concretely introduced. With a time series dataset, in this work, we exploit several LSTM based machine learning models, which are introduced in details as the following.

1) LSTM Cell: LSTM is developed from recurrent neural networks (RNNs), which is a kind of artificial neural networks (ANNs) with recurrent connections. For ANNs, one key assumption is that all the outputs or inputs are independent to each other. However, with the help of recurrent connections, a RNN is able to execute same task for every element from a sequence with the output being depended on the previous computations, which makes it suitable for utilizing sequential data to perform sequence recognition or prediction problem [35]. Whereas, one drawback when a RNN performs back-propagation to optimize the parameters is the gradient vanishing or gradient explosion problem due to the sequential multiplication of $\tanh'$ [36]. Aiming at solving this problem, LSTM is proposed through introducing the gates. The structure of one LSTM cell consists of three main parts: forget gate, input gate, and output gate, which is illustrated in Fig. 3.

At each time t, the calculation equations are as follows,

$$f^{(t)} = \sigma(\omega_f(h^{(t-1)}, x^{(t)}) + b_f), \tag{10}$$

$$i^{(t)} = \sigma(\omega_i(h^{(t-1)}, x^{(t)}) + b_i), \tag{11}$$

$$\widetilde{C}^{(t)} = \tanh(\omega_c(h^{(t-1)}, x^{(t)}) + b_c), \tag{12}$$

$$o^{(t)} = \sigma(\omega_o(h^{(t-1)}, x^{(t)}) + b_o), \tag{13}$$

$$h^{(t)} = \tanh(c^{(t)}) * o^{(t)}, \tag{14}$$

where σ is sigmoid function; $f^{(t)}$, $i^{(t)}$, $o^{(t)}$ are the value of forget gate, input gate and output gate, respectively, and the corresponding ω and b are weight and bias; $h^{(t)}$ is the hidden state. When performing back-propagation among LSTM, we can obtain

$$\begin{aligned}\frac{\partial C^{(t)}}{\partial C^{(t-1)}} &= C^{(t-1)}\sigma'(\cdot)\omega_f * o^{(t-1)}\tanh'(C^{(t-1)}) \\ &+ \widetilde{C}^{(t)}\sigma'(\cdot)\omega_i * o^{(t-1)}\tanh' \\ &+ i^{(t)}\tanh'(\cdot) * o^{(t-1)}\tanh'(C^{(t-1)}) + f^{(t)}.\end{aligned} \tag{15}$$

For k steps back-propagation, the derivative shown in (15) will be multiplied over k times. If the LSTM architecture is sufficiently large, e.g. $k \to \infty$, when the network starts approaching to converge to zero, we can adjust the value of $\frac{\partial C^{(t)}}{\partial C^{(t-1)}}$ closer to one, like 0.97, through controlling the output value of forget gate $f^{(t)}$ [37]. In this way, the gradient vanishing problem can be avoided, which is also the reason for why we only focus on those LSTM based models in this work.

2) BiLSTM: BiLSTM is inspired by the directional RNN and is proposed by [38]. Conventional RNN or LSTM processes series data in a forward direction or in time order. However, for BiLSTM or bidirectional RNN, both forward and backward direction information are utilized to process the sequence data through two independent LSTM or RNN layers [39]. The structure of the BiLSTM network is illustrated in Fig. 4.

As we can see, the overall structure of a BiLSTM can be divided into four layers: input layer, forward layer, backward layer, and output layer. The functionality of input layer is intuitive, which feeds the series data into the network. The forward layer calculates $\overrightarrow{h^{(t)}}$ chronologically, in other words, from $t = 0$ to $t = N$. While the backward layer calculates $\overleftarrow{h^{(t)}}$ unchronologically, i.e., from $t = N$ to $t = 0$. For both forward and backward layers, LSTM is the basic element for configuration. All the formulations and structures inside these LSTM cells are totally the same as what is discussed in Subsection IV-B1. The output layer utilizes the output of forward and backward layers to calculate the current output through a sigmoid activation function. Therefore, unlike the output of LSTM in (13), the final output of the BiLSTM network can be expressed as

$$o^{(t)} = \sigma\big(\omega_o\big(x^{(t)}, \overrightarrow{h^{(t)}}, \overleftarrow{h^{(t)}}\big) + b_o\big). \tag{16}$$

3) Stacked LSTM and Stacked BiLSTM: In the field of machine learning, there is a kind of model named as deep neural networks (DNNs), which is developed from ANNs. Generally, compared with ANNs, a DNN is possessed of a deep architecture, which has more than one hidden layers. With multiple layers, a DNN is able to extract high-level and more essential representations so as to fitting a high-dimensional non-linear

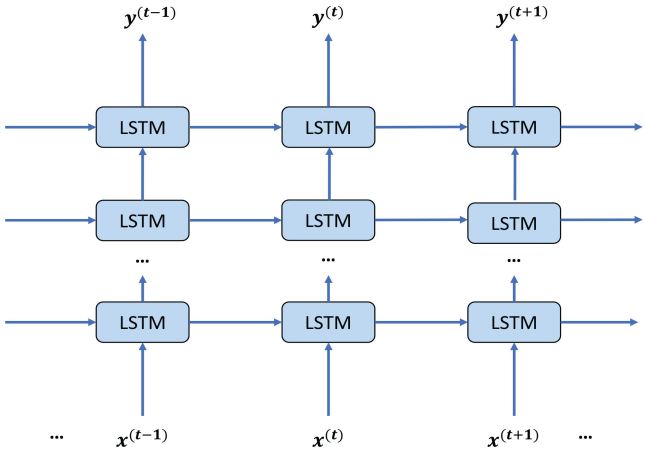


Fig. 5. The structure of a stacked LSTM network.

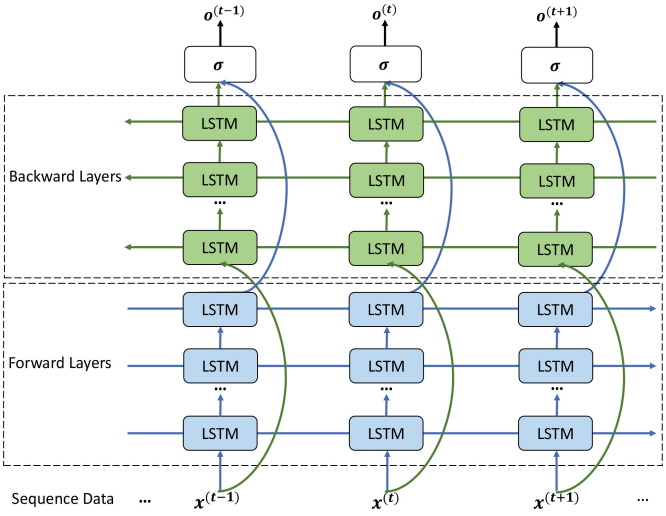


Fig. 6. The structure of a Stacked BiLSTM network.

model [40]. This idea works for LSTM and BiLSTM as well. Based on the shallow structure, a deeper model can be built by adding more hidden layers. The names of deep LSTM and BiLSTM model are stacked LSTM and stacked BiLSTM, respectively, and the corresponding architectures are shown in Fig. 5 and Fig. 6, respectively.

The architecture becomes hierarchical paradigm and the output function needs some modifications accordingly. For stacked LSTM, in layer l, the input is the ouput of the last layer, which can be described as

$$h_l^{(t)} = \omega_{(l-1,l)} h_{l-1}^{(t)} + b_l. \tag{17}$$

Suppose the number of hidden layers is M, the output of the network is

$$o^{(t)} = \sigma(\omega_o(h_M^{(t)}) + b_o). \tag{18}$$

Similarly, the output of a stacked BiLSTM network can be expressed as

$$o^{(t)} = \sigma\big(\omega_o\big(\overrightarrow{h_M^{(t)}}, \overleftarrow{h_M^{(t)}}\big) + b_o\big). \tag{19}$$

Fig. 7. The 3D model of Manhattan area built in Unity.

Although a deeper network can fit a complex non-linear model well, it also easily brings over-fitting problem. Over-fitting means the model is exactly suitable for the training dataset while performs poorly upon test dataset. To tackle with this problem, one effective method is to adopt the dropout strategy. The main idea of dropout is to randomly ignore some hidden LSTM units by a certain percentage during the training phase. The selected hidden units will not update the parameters in the back-propagation process. But for the other units, they will optimize the parameters normally according to the back propagated error. By this way, the model can never fit the training dataset perfectly so as to avoid the over-fitting problem [41]. Therefore, based on these two deep models, we add dropout strategy during the training phase, which are named as S-LSTM with dropout (S-LSTM-D) and S-BiLSTM with dropout (S-BiLSTM-D), respectively.

V. Simulation Results

A. Data Generation

There are many different edge computing resources in reality, such as computation memory, computation power, etc. In this work, we focus on the computation memory consumption. In order to obtain the memory utilization data in diverse traffic situation, we build our own simulation environment through implementing the game engine Unity3D [42]. We build up the 3D model of real-world Manhattan area as the geographical background, which is shown in Fig. 7, where the red dots indicate the locations of edge servers and the green dashed boxes represent the simulation areas. The traffic AI package is implemented to yield traffic flows. Vehicle models are generated in specified areas and move forward following the pre-defined road network obeying traffic rules. In addition, we adjust the total number of vehicles appear in the roadside unit (RSU) coverage area to simulate peak period and off-peak period traffic.

Edge server model and vehicle receiver model are built to simulate edge computing assisted computation execution

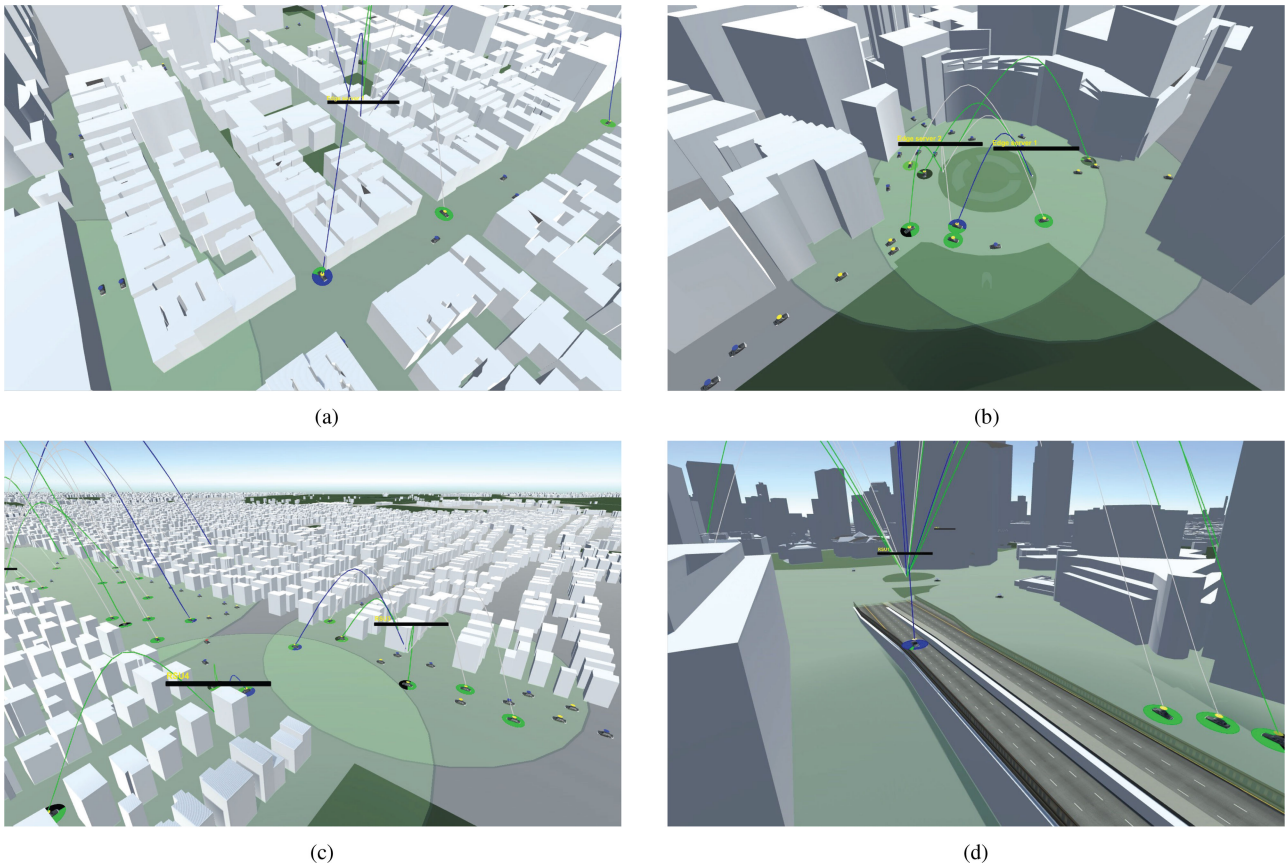


Fig. 8. The different scenarios built in Unity. (a) The multi-intersection scenario. (b) The roundabout scenario. (c) The highway scenario. (d) The bridge scenario.

behavior on both units when vehicle receiver is located within the connection range of edge server. Every offloaded computation task will consume a certain amount of memory in the edge server during the period of the task execution. Therefore, we can obtain the memory utilization data by documenting the memory consumption of the edge server. Memory consumption sizes are defined in the vehicle receiver model to simulate offloading computation with different data size. Accordingly, a larger data size will occupy more memory capacity on edge server, resulting in taking longer time to be computed or processed, and vice versa. Vehicle receiver model is attached to the vehicle model of the traffic AI package, and edge server models are placed in the area, where is considered to perform simulated computation offloading service for all vehicles within the range of the edge server.

For the purpose of enriching the scenarios, in this work, we choose four different road maps to simulate traffic flow, i.e., multiple intersections, roundabout, highway, and bridge areas, which is illustrated in Fig. 8. Besides, the simulation video is available at [43]. As is shown in Fig. 8, edge servers are located right below the horizontal black bars, surrounded by a green circle representing the coverage range of each edge server. The spline connecting vehicles and edge servers represents the connection link, with different color represents different ongoing operation (green for uploading, white for processing, and blue for downloading). The small dot on top of the vehicles is the vehicle receiver model, and the color of the dot stands for the status of the receiver (blue for task complete, yellow for task in process, and red for disconnected). Besides, the larger green or blue disk attached to the bottom of vehicles represents upload and download progress. Apart from creating different scenarios, we also vary the memory consumption (MC) size in vehicle receiver model and number of vehicles utilizing the driving AI package. The number of cars for multi-intersection, roundabout, highway, and bridge are set as {30, 60, 100}, {20, 35, 50}, {30, 60, 100}, and {45, 90, 150}, respectively. For the small data size situation, the MC will be randomly generated within the range of [18, 22] (MB). And for big data size situation, the MC will be randomly generated within the rage of [36, 44] (MB). Therefore, for each road scenario, there are six dataset in total with the combination of different data size and number of cars and these three factors constitute the meta-feature space $\mathcal{F}$. For convenience, we name the dataset in the format "Scenario-number of vehicles-datasize". For example, Roundabout-20-B means the data is generated in roundabout scenario with 20 vehicles and the data size is big. The details of settings for variation of situations can be summarized in Table II.

B. Simulation and Evaluation

For the performance evaluation metrics space $\mathcal{E}$, we adopt the suggestions in [44] and use three measurements totally, i.e. root

TABLE II
PARAMETERS SETTING FOR DIFFERENT ROAD MAPS

Road Map	Amount of Vehicles			MC Range (MB)	
	Small	Medium	Large	Small	Large
Multi-Intersections	30	60	100	[18,22]	[36,44]
Roundabout	20	35	50	[18,22]	[36,44]
Highway	30	60	100	[18,22]	[36,44]
Bridge	45	90	150	[18,22]	[36,44]

TABLE III
PARAMETERS SETTINGS FOR $\mathcal{M}$

Models	Hidden Layer	Units	Dropout Percentage
LSTM	1	64	-
BiLSTM	1	64	-
S-LSTM	3	32-32-32	-
S-BiLSTM	2	32-32	-
S-LSTM-D	3	32-64-32	10%
S-BiLSTM-D	2	64-64	10%

mean square error (RMSE), mean absolute percentage (MAP), and mean GEH (MGEH), whose definitions are

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum\nolimits_{i=1}^{n}\left(y_i - Y_i\right)^2}, \tag{20}$$

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - Y_i|}{Y_i}, \tag{21}$$

$$\text{MGEH} = \frac{1}{n}\sum_{i=1}^{n}\sqrt{\frac{2\left(y_i - Y_i\right)^2}{y_i + Y_i}}, \tag{22}$$

respectively, where y_i is the predicted value and Y_i is the ground truth. Therein RMSE and MAPE are widely used statistical metrics. While GEH is named by the initials of creator, Geoffrey E. Havers, and has no special mathematical or statistical meaning. Even so, GEH has been approved to be an effective measurement for a variety of traffic analysis purpose and a smaller GEH value indicates a better regression of observed flows [45]. In addition, as is discussed in Section IV-B, $\mathcal{M}$ contains six models, i.e., LSTM, BiLSTM, S-LSTM, S-BiLSTM, S-LSTM-D, and S-BiLSTM-D. The parameters settings are as shown in Table III. The simulations are performed under TensorFlow framework and the GPU version is NVIDIA 1080Ti. The results are shown in Table IV, which forms our experience dataset $\mathcal{D}$.

As we can see in Table IV, for some scenarios, MAPE is not applicable because there are zeros in the dataset, resulting in the infinite value. Among the three metrics, it is obvious that they keep consistent to each other actually. Checking throughout the table, one conclusion is that LSTM performs better in the majority cases instead of those deep models. Generally, deep network is supposed to work better than shallow networks, like the performance difference between DNNs and ANNs, however, which is not the case here. The main difference is, in DNNs or ANNs, a vital assumption is the data samples are independent to each other. Whereas, for time series data, one significant property is time dependency, which means the information comes from previous LSTM makes more sense than the information transmitted between hidden layers. In addition, in the stacked architecture, if the previous layer has already made wrong prediction, the next layer will continue forecasting based on the incorrect results, which means errors will be transmitted and enlarged. Besides, time dependency can also explain why the dropout strategy is not suitable here. Since the information that each LSTM brings may be of great importance for cells in the later series. Dropout strategy will erase the randomly selected units so that anticipation cannot be calculated accurately. Moreover, for time series data, the input can be as few as even two dimensional, i.e., time and value. Unlike high dimensional cases, overfitting is not that common. Also, comparing the bidirectional model with unbidirectional model, generally, the performances are similar. But there are still some numerical differences between them, which is because if the forward information is different from backward information, it is easy to introduce bias in the output phase.

Based on the results in Table IV, we generate the data for model selection recommendations. The data is four-dimensional. The first three features are roadmap, the number of vehicles, and memory consumption size. And the rest one is the suggested model. With the foundation of this dataset $\mathcal{D}$, we build a two-hidden layer DNN to perform the model selection problem, or we can say, a classification problem. The hidden units in each hidden layer is 4. The loss function is defined as MSE as well and the gradient descent methods is also Adam. In addition, according to the evaluation from [46], the peak time of traffic happens averagely from 6am to 9am and 4pm to 7pm. Therefore, when we assess the performance on a roadmap basis and quantitative analysis, we add different weights to the values. The datasets with a larger number of vehicles will get 25%, as is regarded as peak time. The two hours around peak time are defined for medium number of vehicles, which is 33%. The rest 42% goes for the datasets with small amount of vehicles, which is considered as off-peak time. In addition, for the percentage of different memory consumption size, it is divided equally. Finally, the final scores obtained by our meta-learning method and the non-meta methods are summarized in Table V, which is statistically calculated among the different roadmaps. Obviously, we can find that our proposed method always achieves the best evaluation scores among all the roadmaps.

Besides, we conduct the quantitative analysis among all the methods as well. Here, to calculate cost defined in (1) and waste defined in (2), we take the price of the AWS as reference. According to the data and description from reference [12], the one year of all upfront price is 541 dollars and can save 43% compared with paying on demand. Here, we just do a simple normalization and define the reservation unit price as 1.48 dollars and the unit price for real-time request as 2.60 dollars, respectively. The concrete details for the cost are summarized in Fig. 9. Also, we add the best case and the worst case as the benchmark values, which indicates all the volumes are purchased by reservation and real-time requests, respectively.

Obviously, we can find that although there is gap between the cost of our proposed meta-learning method and the counterpart of the best case, the proposed method always gives the better price than all the other methods, which helps save up to 39.93%, 37.15%, 5.62%, and 70.47% for the multi-intersections,

TABLE IV
SUMMARY OF RESULTS FOR DIFFERENT SCENARIOS

Dataset	Multi-Intersections-30-B			Roundabout-20-B			Highway-30-B			Bridge-45-B		
Model	RMSE	MAPE	MEGH	RMSE	MAPE	MEGH	RMSE	MAPE	MEGH	RMSE	MAPE	MEGH
LSTM	33.18	13.14	1.80	19.11	-	1.41	11.25	-	1.18	16.57	16.38	1.40
BiLSTM	50.07	21.26	2.70	32.64	-	2.54	30.63	22.14	2.06	19.52	24.76	1.63
S-LSTM	43.19	17.81	2.34	43.42	-	3.34	38.56	23.90	2.65	47.64	72.08	3.73
S-BiLSTM	49.39	19.90	2.60	33.84	-	2.65	50.82	39.15	3.40	23.63	-	2.57
S-LSTM-D	43.11	16.50	2.30	31.17	-	2.35	52.65	38.73	3.51	20.91	24.99	1.81
S-BiLSTM-D	42.78	17.34	2.34	35.98	-	2.80	34.49	20.66	2.16	23.36	28.93	2.01
Dataset	Multi-Intersections-30-S			Roundabout-20-S			Highway-30-S			Bridge-45-S		
Model	RMSE	MAPE	MEGH	RMSE	MAPE	MEGH	RMSE	MAPE	MEGH	RMSE	MAPE	MEGH
LSTM	15.87	-	1.74	11.18	-	1.63	11.25	-	1.18	15.78	15.28	1.30
BiLSTM	16.70	-	1.93	20.19	-	3.15	13.03	-	1.54	19.52	24.76	1.63
S-LSTM	28.24	-	2.87	15.45	-	2.44	19.86	-	2.33	17.32	15.65	1.42
S-BiLSTM	24.12	-	2.76	20.84	-	3.41	26.63	-	2.57	19.03	19.07	1.63
S-LSTM-D	17.85	-	1.98	14.67	-	2.31	19.78	-	1.93	21.45	21.59	1.86
S-BiLSTM-D	17.54	-	1.93	18.95	-	2.87	24.75	-	2.50	20.03	20.16	1.65
Dataset	Multi-Intersections-60-B			Roundabout-35-B			Highway-60-B			Bridge-90-B		
Model	RMSE	MAPE	MEGH	RMSE	MAPE	MEGH	RMSE	MAPE	MEGH	RMSE	MAPE	MEGH
LSTM	82.85	6.87	2.26	30.03	8.93	1.37	121.37	11.79	3.70	74.01	7.27	2.19
BiLSTM	58.32	4.98	1.55	28.00	8.11	1.26	102.04	10.15	3.17	105.20	9.88	2.93
S-LSTM	155.58	12.87	4.34	45.86	13.68	2.14	180.54	18.10	5.78	46.09	3.96	1.21
S-BiLSTM	170.55	14.98	4.57	66.48	25.44	3.57	115.20	11.14	3.50	64.89	6.11	1.85
S-LSTM-D	81.27	6.06	2.01	36.16	11.50	1.76	148.55	15.05	4.76	65.95	5.93	1.77
S-BiLSTM-D	75.08	5.88	1.93	59.11	22.80	3.27	187.68	18.84	6.04	120.86	11.48	3.40
Dataset	Multi-Intersections-60-S			Roundabout-35-S			Highway-60-S			Bridge-90-S		
Model	RMSE	MAPE	MEGH	RMSE	MAPE	MEGH	RMSE	MAPE	MEGH	RMSE	MAPE	MEGH
LSTM	16.53	11.24	1.18	10.64	-	1.10	15.50	-	1.47	82.30	6.85	2.27
BiLSTM	27.46	19.31	1.93	23.34	-	2.64	46.17	-	5.46	165.65	14.34	4.85
S-LSTM	20.43	14.86	1.48	23.13	-	3.12	24.79	-	2.41	161.81	13.77	4.67
S-BiLSTM	19.77	13.16	1.38	25.62	-	3.20	23.66	-	2.24	127.81	10.79	3.63
S-LSTM-D	38.64	31.30	2.99	17.48	-	2.48	25.46	-	2.56	247.57	21.35	7.41
S-BiLSTM-D	24.55	19.33	1.88	21.45	-	2.95	21.11	-	1.98	152.22	13.04	4.41
Dataset	Multi-Intersections-100-B			Roundabout-50-B			Highway-100-B			Bridge-150-B		
Model	RMSE	MAPE	MEGH	RMSE	MAPE	MEGH	RMSE	MAPE	MEGH	RMSE	MAPE	MEGH
LSTM	108.03	4.11	1.88	40.49	8.07	1.64	48.38	2.47	0.94	52.59	2.94	1.11
BiLSTM	289.61	12.07	5.45	119.53	25.64	4.98	56.02	2.90	1.10	71.93	4.22	1.59
S-LSTM	354.29	15.20	6.79	92.69	19.35	3.81	98.95	5.40	2.07	52.10	2.93	1.11
S-BiLSTM	89.52	3.19	1.47	73.26	14.86	2.90	65.72	3.62	1.43	43.53	2.33	0.89
S-LSTM-D	352.13	14.96	6.67	105.73	21.54	4.15	68.31	3.71	1.48	52.65	2.88	1.12
S-BiLSTM-D	224.16	8.55	3.83	83.02	17.18	3.34	93.49	5.21	2.01	145.20	8.96	3.58
Dataset	Multi-Intersections-100-S			Roundabout-50-S			Highway-100-S			Bridge-150-S		
Model	RMSE	MAPE	MEGH	RMSE	MAPE	MEGH	RMSE	MAPE	MEGH	RMSE	MAPE	MEGH
LSTM	32.59	4.28	1.06	12.98	10.24	0.96	58.96	8.21	2.06	108.74	5.49	2.35
BiLSTM	42.63	5.98	1.46	31.29	32.47	2.66	30.97	4.45	1.04	105.76	4.74	1.93
S-LSTM	183.62	25.54	7.10	27.06	25.25	2.12	188.18	27.18	7.28	196.14	9.96	4.35
S-BiLSTM	49.06	6.51	1.56	38.93	39.93	3.19	99.96	14.61	3.72	241.47	12.49	5.48
S-LSTM-D	87.74	9.87	2.69	49.38	53.17	4.21	213.93	30.79	8.37	224.48	11.49	5.04
S-BiLSTM-D	39.35	5.07	1.28	30.21	28.36	2.82	47.77	6.18	1.52	169.66	8.66	3.76

TABLE V
SCORES OBTAINED BY PROPOSED META METHOD AND NON-META METHODS

Roadmap	Multi-Intersections			Roundabout			Highway			Bridge		
Model	RMSE	MAPE	MEGH	RMSE	MAPE	MEGH	RMSE	MAPE	MEGH	RMSE	MAPE	MEGH
Meta-learning	37.91	9.13	1.51	19.42	8.56	1.35	37.23	11.64	1.63	46.63	9.31	1.49
LSTM	44.26	9.56	1.68	19.76	9.03	1.37	43.92	12.65	2.72	52.63	10.03	1.72
BiLSTM	69.71	11.78	2.41	38.42	17.14	2.79	44.50	13.57	2.45	74.45	13.67	2.36
S-LSTM	111.28	17.15	3.79	38.71	17.40	2.82	60.79	20.08	3.57	78.98	22.96	2.73
S-BiLSTM	64.16	14.21	2.49	40.70	26.28	3.15	59.26	22.41	2.84	75.99	14.63	2.42
S-LSTM-D	87.57	16.20	2.89	37.87	22.64	2.72	79.20	25.55	3.58	95.27	16.08	3.06
S-BiLSTM-D	62.05	13.14	2.16	38.98	22.79	2.99	64.55	16.32	2.74	93.53	16.56	2.76

roundabout, highway, and bridge scenarios, respectively, compared with the fully real time requests. Apart from the cost, we also calculate the amount of waste. Actually, the total cost is made up by two parts: one is the reservation part and the other one is the real-time request when we have insufficient reservation. Intuitively, both of them can be the source of waste. If the amount of reservation is more than that of needed, it means we make excess reservation, which leads to a waste of money for the customer and a waste of computation or storage resources for edge platform as well. Similarly, if the

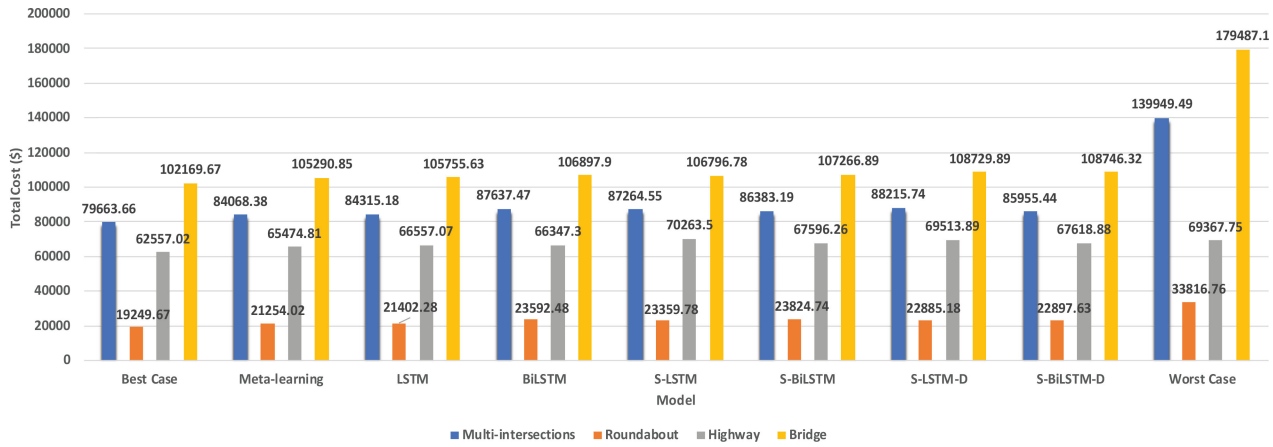


Fig. 9. Total cost for different methods.

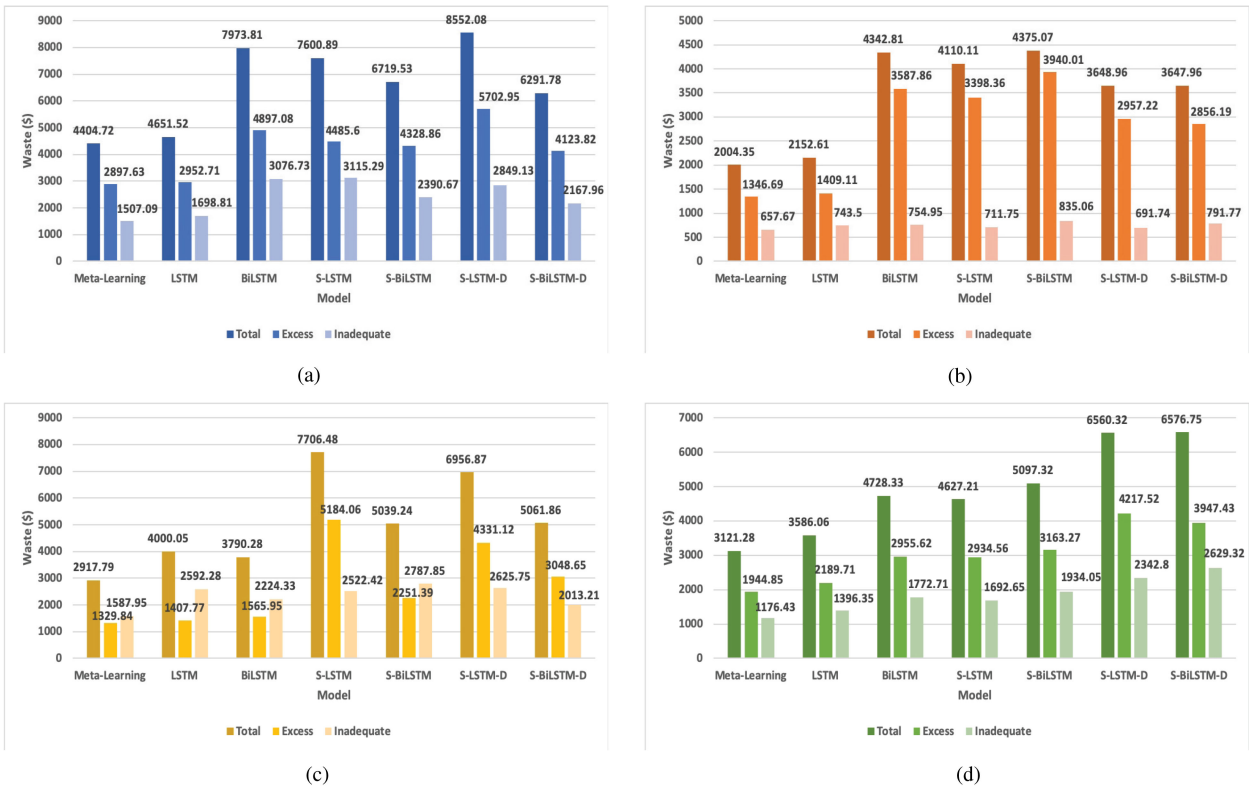


Fig. 10. Waste for different methods. (a) Waste in multi-intersection scenario. (b) Waste in roundabout scenario. (c) Waste in highway scenario. (d) Waste in bridge scenario.

amount of reservation is less than the demands, it indicates we make inadequate reservation, which means purchasing more resources is inevitable for meeting the demands with a relatively expensive price. Therefore, when we calculate the waste, the distinguishment for these two kinds of waste is also took into consideration, and the details are summarized in Fig. 10. For the total waste, the results keep consistent with the results of cost, i.e., the method gives the most economical scheme brings the least waste. Meanwhile, no matter for the excess waste or the inadequate waste, we can find that our proposed meta-learning gives out the least waste.

VI. Conclusion

In order to minimize the expenses to consume edge computing resources, this paper proposed a two-stage meta-learning approach. In the first stage, a DNN is utilized to learn the experience dataset so as to figure out which machine learning model performs better in a specific situation. In the second stage, the resource amount anticipation will be conducted by the machine learning model selected by the DNN obtained in the first stage according to the meta-features. In addition, due to the fact that there is no open edge computing based vehicular network

dataset, we program in the game engine Unity3D to build the 3D model of Manhattan area as in real world. Meanwhile, we adjust the factors like different roadmaps, the number of vehicles, and the randomness of memory consumption sizes for the traffic, which makes our data get closer to the practice. Eventually, we find that out proposed meta-learning method always gives the most economical predictions, which helps save up to 39.93%, 37.15%, 5.62%, and 70.47% for multi-intersections, roundabout, highway, and bridge scenarios, respectively, compared with the fully real time requests.

REFERENCES

[1] National Highway Traffic Safety Administration (NHTSA). Automated vehicles for safety, 2019. [Online]. Available: https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety#issue-road-self-driving

[2] Intel. Data is the new oil in the future of automated driving, 2016. [Online]. Available: https://newsroom.intel.com/editorials/krzanich-the-future-of-automated-driving/#gs.jpzzzs

[3] D. Chen, X. Zhang, L. L. Wang, and Z. Han, "Prediction of cloud resources demand based on hierarchical pythagorean fuzzy deep neural network," *IEEE Trans. Services Comput.*, to be published, doi: 10.1109/TSC.2019.2906901.

[4] X. Yang, Z. Fei, J. Zheng, N. Zhang, and A. Anpalagan, "Joint multi-user computation offloading and data caching for hybrid mobile cloud/edge computing," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 11 018–11 030, Nov. 2019.

[5] H. Guo, J. Liu, J. Zhang, W. Sun, and N. Kato, "Mobile-edge computation offloading for ultradense IoT networks," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4977–4988, Dec. 2018.

[6] Z. Zhou, J. Feng, Z. Chang, and X. Shen, "Energy-efficient edge computing service provisioning for vehicular networks: A consensus ADMM approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 5087–5099, May 2019.

[7] A. Ndikumana *et al.*, "Joint communication, computation, caching, and control in big data multi-access edge computing," *IEEE Trans. Mobile Comput.*, vol. 19, no. 6, pp. 1359–1374, Jun. 1, 2020.

[8] H. Guo, J. Zhang, and J. Liu, "FiWi-enhanced vehicular edge computing networks: Collaborative task offloading," *IEEE Veh. Technol. Mag.*, vol. 14, no. 1, pp. 45–53, Mar. 2019.

[9] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7944–7956, Aug. 2019.

[10] H. Guo, J. Liu, and J. Lv, "Toward intelligent task offloading at the edge," *IEEE Netw.*, vol. 34, no. 2, pp. 128–134, Mar./Apr. 2020.

[11] B. Antonio, F. Stefano, and I. Ahmad, "Deploying fog applications how much does it cost, by the way?" in *Proc. 7th Int. Conf. Cloud Comput. Services Sci.*, Porto, Portugal, Apr. 2017, pp. 68–77.

[12] Amazon Web Services, "AWS Pricing—How does AWS pricing work?", 2006. [Online]. Available: https://aws.amazon.com/pricing/?nc1=h_ls

[13] J. Xu, R. Rahmatizadeh, L. Bölöni, and D. Turgut, "Real-time prediction of taxi demand using recurrent neural networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 8, pp. 2572–2581, Aug. 2018.

[14] C. Lemke, M. Budka, and B. Gabrys, "Metalearning: A survey of trends and technologies," *Artif. Intell. Rev.*, vol. 44, no. 1, pp. 117–130, Jun. 2015.

[15] S. Yu, R. Langar, X. Fu, L. Wang, and Z. Han, "Computation offloading with data caching enhancement for mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11 098–11 112, Nov. 2018.

[16] Q. Fan and N. Ansari, "On cost aware cloudlet placement for mobile edge computing," *IEEE/CAA J. Automatica Sinica*, vol. 6, no. 4, pp. 926–937, Jul. 2019.

[17] H. Wang, B. Kim, J. Xie, and Z. Han, "E-auto: A communication scheme for connected vehicles with edge-assisted autonomous driving," in *Proc. IEEE Int. Conf. Commun.*, Shanghai, China, May 2019, pp. 1–6.

[18] J. Liu, H. Guo, J. Xiong, N. Kato, J. Zhang, and Y. Zhang, "Smart and resilient EV charging in SDN-enhanced vehicular edge computing networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 1, pp. 217–228, Jan. 2020.

[19] Y. Liu, H. Yu, S. Xie, and Y. Zhang, "Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 11 158–11 168, Nov. 2019.

[20] T. K. Rodrigues, K. Suto, H. Nishiyama, J. Liu, and N. Kato, "Machine learning meets computation and communication control in evolving edge and cloud: Challenges and future perspective," *IEEE Commun. Surv. Tut.*, vol. 22, no. 1, pp. 38–67, Jan.-Mar. 2020.

[21] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang, "Learning-based computation offloading for IoT devices with energy harvesting," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1930–1941, Feb. 2019.

[22] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—A key technology towards 5 g," *Eur. Telecommun. Standards Institute White Paper*, vol. 11, no. 11, pp. 1–16, Sep. 2015.

[23] P. Gupta, A. Seetharaman, and J. R. Raj, "The usage and adoption of cloud computing by small and medium businesses," *Int. J. Inf. Manage.*, vol. 33, no. 5, pp. 861–874, Oct. 2013.

[24] J. M. García, P. Fernández, A. Ruiz-Cortes, S. Dustdar, and M. Toro, "Edge and cloud pricing for the sharing economy," *IEEE Internet Comput.*, vol. 21, no. 2, pp. 78–84, Mar. 2017.

[25] A. Mazrekaj, I. Shabani, and B. Sejdiu, "Pricing schemes in cloud computing: An overview," *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 2, pp. 80–86, Feb. 2016.

[26] B. Li, W. Xie, W. Zeng, and W. Liu, "Learning to update for object tracking with recurrent meta-learner," *IEEE Trans. Image Process.*, vol. 28, no. 7, pp. 3624–3635, Feb. 2019.

[27] S. Canuto, D. X. Sousa, M. A. Gonçalves, and T. C. Rosa, "A thorough evaluation of distance-based meta-features for automated text classification," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 12, pp. 2242–2256, Mar. 2018.

[28] D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales, "Learning to generalize: Meta-learning for domain generalization," in *Proc. 32th AAAI Conf. Artif. Intell.*, New Orleans, LA, 2018, pp. 3490–3497.

[29] M. Vartak, A. Thiagarajan, C. Miranda, J. Bratman, and H. Larochelle, "A meta-learning perspective on cold-start recommendations for items," in *Proc. 31th Conf. Neural Inf. Process. Syst.*, Long Beach, CA, 2017, pp. 6904–6914.

[30] J. R. Rice, "The algorithm selection problem," in *Advances in Computers*. Elsevier, 1976, vol. 15, pp. 65–118.

[31] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, Apr. 1997.

[32] D. G. Ferrari and L. N. De Castro, "Clustering algorithm selection by meta-learning systems: A new distance-based problem characterization and ranking combination methods," *Inf. Sci.*, vol. 301, pp. 181–194, Apr. 2015.

[33] J. Vanschoren, "Meta-learning: A survey," 2018, *arXiv:1810.03548*.

[34] M. Tripathy and A. Panda, "A study of algorithm selection in data mining using meta-learning." *J. Eng. Sci. Technol. Rev.*, vol. 10, no. 2, pp. 51–64, Mar. 2017.

[35] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee, "Recent advances in recurrent neural networks," 2017, *arXiv:1801.01078*.

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

[37] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proc. Int. Conf. Mach. Learn.*, Atlanta, GA, Jun. 2013, pp. 1310–1318.

[38] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural Netw.*, vol. 18, no. 5-6, pp. 602–610, Jul. 2005.

[39] Z. Cui, R. Ke, and Y. Wang, "Deep bidirectional and unidirectional lstm recurrent neural network for network-wide traffic speed prediction," 2018, *arXiv:1801.02143*.

[40] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[41] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jun. 2014.

[42] Unity. Unity for games: Create a world with more play, 2019. [Online]. Available: https://unity.com/solutions/game

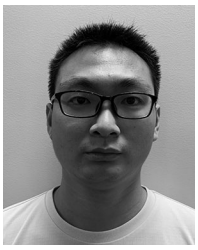
[43] Video for "Edge computing resources reservation in vehicular networks: A meta-learning approach", 2019. [Online]. Available: https://youtu.be/aiEpy1eV7pw

[44] J. Mackenzie, J. F. Roddick, and R. Zito, "An evaluation of htm and lstm for short-term arterial traffic flow prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 5, pp. 1847–1857, Aug. 2019.

[45] Transport for London. Traffic modeling guidelines version 3.0, 2010. [Online]. Available: http://content.tfl.gov.uk/traffic-modelling-guidelines.pdf

[46] T. D. Wemegah, S. Zhu, and C. Atombo, "Modeling the effect of days and road type on peak period travels using structural equation modeling and big data from radio frequency identification for private cars and taxis," *Eur. Transport Res. Rev.*, vol. 10, no. 2, pp. 1–14, Jun. 2018.

Dawei Chen (Student Member, IEEE) received the B.S. degree in telecommunication engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2015. He is currently working toward the Ph.D. degree with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX, USA. His research interests include machine learning, edge computing, and wireless networks.



Yin-Chen Liu received the bachelor's degree in mechanical engineering from National Taipei University of Technology, Taipei, Taiwan, in 2013 and the M.S. degree in mechanical engineering from the University of California, Riverside, Riverside, CA, USA, in 2017. He is currently a Research Scientist with Toyota Motor North America Info Tech Labs, Mountain View, CA, USA. His research interests include cyber physical system, robotics control, edge computing and optimization.

BaekGyu Kim (Member, IEEE) received the Ph.D. degree in computer science from the University of Pennsylvania, Philadelphia, PA, USA. He is currently a Principal Researcher with Toyota Motor North America, InfoTech Labs, Mountain View, CA, USA. His research interests include software platform technologies for connected cars, and model based software development for high-assurance systems.

Jiang Xie (Fellow, IEEE) received the B.E. degree from Tsinghua University, Beijing, China, the M.Phil. degree from The Hong Kong University of Science and Technology, Hong Kong, and the M.S. and Ph.D. degrees from Georgia Institute of Technology, Atlanta, GA, USA, all in electrical and computer engineering. She joined the Department of Electrical and Computer Engineering, University of North Carolina at Charlotte (UNC-Charlotte), Charlotte, NC, USA, as an Assistant Professor, in August 2004, where she is currently a Full Professor. Her current research interests include resource and mobility management in wireless networks, mobile computing, Internet of Things, and cloud/edge computing. She is on the Editorial Boards for the IEEE/ACM TRANSACTIONS ON NETWORKING and *Journal of Network and Computer Applications* (Elsevier). She received the US National Science Foundation (NSF) Faculty Early Career Development (CAREER) Award in 2010, a Best Paper Award from IEEE Global Communications Conference (Globecom 2017), a Best Paper Award from IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2010), and a Graduate Teaching Excellence Award from the College of Engineering at UNC-Charlotte in 2007. She is a Senior Member of the ACM.

Choong Seon Hong (Senior Member, IEEE) received the B.S. and M.S. degrees in electronic engineering from Kyung Hee University, Seoul, South Korea, in 1983 and 1985, respectively, and the Ph.D. degree from Keio University, Tokyo, Japan, in 1997. In 1988, he joined KT, Gyeonggi-do, South Korea, where he was involved in broadband networks as a member of the Technical Staff. Since 1993, he has been with Keio University. He was with the Telecommunications Network Laboratory, KT, as a Senior Member of Technical Staff and as the Director of the Networking Research Team until 1999. Since 1999, he has been a Professor with the Department of Computer Science and Engineering, Kyung Hee University. His research interests include future internet, intelligent edge computing, network management, and network security. He is a member of the Association for Computing Machinery (ACM), the Institute of Electronics, Information and Communication Engineers (IEICE), the Information Processing Society of Japan (IPSJ), the Korean Institute of Information Scientists and Engineers (KIISE), the Korean Institute of Communications and Information Sciences (KICS), the Korean Information Processing Society (KIPS), and the Open Standards and ICT Association (OSIA). He has served as the General Chair, the TPC Chair/Member, or an Organizing Committee Member of international conferences, such as the Network Operations and Management Symposium (NOMS), International Symposium on Integrated Network Management (IM), Asia-Pacific Network Operations and Management Symposium (APNOMS), End-to-End Monitoring Techniques and Services (E2EMON), IEEE Consumer Communications and Networking Conference (CCNC), Assurance in Distributed Systems and Networks (ADSN), International Conference on Parallel Processing (ICPP), Data Integration and Mining (DIM), World Conference on Information Security Applications (WISA), Broadband Convergence Network (BcN), Telecommunication Information Networking Architecture (TINA), International Symposium on Applications and the Internet (SAINT), and International Conference on Information Networking (ICOIN). He was an Associate Editor for the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT and the IEEE JOURNAL OF COMMUNICATIONS AND NETWORKS. He currently serves as an Associate Editor for the *International Journal of Network Management* and an Associate Technical Editor for the *IEEE Communications Magazine*.

Zhu Han (Fellow, IEEE) received the B.S. degree in electronic engineering from Tsinghua University, Beijing, China, in 1997, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Maryland, College Park, MD, USA, in 1999 and 2003, respectively. From 2000 to 2002, he was an R&D Engineer of JDSU, Germantown, Maryland. From 2003 to 2006, he was a Research Associate with the University of Maryland. From 2006 to 2008, he was an Assistant Professor with Boise State University, Idaho. He is currently a John and Rebecca Moores Professor with the Department of Electrical and Computer Engineering as well as with the Department of Computer Science, University of Houston, Houston, TX, USA. His research interests include wireless resource allocation and management, wireless communications and networking, game theory, big data analysis, security, and smart grid. He received an NSF Career Award in 2010, the Fred W. Ellersick Prize of the IEEE Communication Society in 2011, the EURASIP Best Paper Award for the *Journal on Advances in Signal Processing* in 2015, IEEE Leonard G. Abraham Prize in the field of communications systems (Best Paper Award in IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS) in 2016, and several best paper awards in IEEE conferences. He was an IEEE Communications Society Distinguished Lecturer from 2015 to 2018, and has been AAAS Fellow since 2019 and ACM Distinguished Member since 2019. He is 1% highly cited researcher since 2017 according to Web of Science.