Meta-Learning Based Network Traffic Prediction for IIoT

Ching Chen, Hsin-Lung Wu, *Member, IEEE*, Pin-Han Ho, *Fellow, IEEE*, and Bor-Shing Lin, *Senior Member, IEEE*

***Abstract*— Industrial Internet of Things (IIoT) is a technique which integrates different sensors and controllers into each phase of industrial production. It enhances production efficiency, improves product quality, and reduces product costs and resource consumption, thereby upgrading traditional industries to a new level of intelligence. Since many industrial devices are connected via the network, it results in an issue to predict network traffic volume for further allocating network resource to these edge devices. Previous works paid attetion to establishing traffic prediction models in the IIoT cloud-side server, which may cause huge computing and memory consumption of the cloud server. The goal of this paper is to address this issue by introducing the Model-Agnostic Meta-Learning (MAML) method to achieve real-time network traffic fine-tuning and prediction on edge devices. In our edge-side network traffic prediction framework consists of two stages. In the first cloud-side meta-learning stage, an MAML-based algorithm learns a good initial parameter vector by the collected time series produced from existing edge devices and outputs it as the initial model weight for the lightweight traffic prediction model deployed in a new edge device of the same type. Then, in the second edge-side fine-tuning stage, the new edge device starts from the learned initial model weight to fine-tune the parameter vector of the lightweight prediction model. With this nice initial parameter vector, the edge device can adapt its data quickly and accomplish the traffic prediction task. In addition, this article also recommends several lightweight models which can be deployed in edge devices for traffic prediction. Throughout an experimental analysis, our proposed MAML-based traffic prediction framework outperforms train-from-scratch-based approach. Moreover, experimental results also demonstrate the advantage of edge-side IIoT network traffic prediction compared to the state-of-the-art cloud-side prediction method.**

***Index Terms*— Industrial Internet of Things (IIoT), network traffic prediction, edge computation, meta learning**

I. Introduction

Manuscript received ???. This work was supported in part by the National Science and Technology Council, Taiwan, under Grant NSTC-112-2221-E-305-004; and in part by National Taipei University under Grant 2023-NTPU-IJRP-No.0002.

Ching Chen and Hsin-Lung Wu are with Department of Computer Science and Information Engineering, National Taipei University, New Taipei City 237303, Taiwan (e-mail: kafuchino0410@gmail.com; hsinlung@mail.ntpu.edu.tw).

Pin-Han Ho is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, On N2L 3G1, Canada (e-mail: p4ho@uwaterloo.ca).

Bor-Shing Lin is with Department of Computer Science and Information Engineering, National Taipei University, New Taipei City 237303, Taiwan (e-mail: bslin@mail.ntpu.edu.tw).

RECENTLY, the Industrial Internet of Things (IIoT) became an important research issue in the field of information and communication technology. It is a system of interconnected devices, instruments, and industrial applications connected by network, which aims at achieving flexible resource allocation and data interconnection. In IIoT, lots of different machines and information systems not only generate a huge amount of data but also connect on Internet and the cloud in a collaborative way. This results in heavy network traffic which is a great challenge for network delay, bandwidth resources, and key communication nodes of IIoT. In recent years, some researches [1] introduced machine learning (ML) techniques to obtain intelligent IIoT communication systems which greatly help to achieve reliable and efficient data transmission among IIoT devices. One of important ML-based IIoT issues is network traffic prediction [2] which can ensure bandwidth efficiency and help energy allocation. Moreover, it is easy for users to manage IIoT devices in order to maintain performance of complex industrial devices by monitoring their network workload. In general, ML-based network traffic prediction can effectively estimate the future demand for communication resources of each IIoT device so that the network resources can be allocated and adjusted efficiently. Most of existing ML-based network traffic prediction algorithms relied on centralized network traffic management systems which are facing many challenges. The main challenges can be summarized as follows [3], [4].

1) Network traffic is the amount of data moving across a computer network at any given time. In an IIoT, both the amount of network traffic and the computational complexity of centralized traffic prediction management algorithms simultaneously increase with the number of interconnected edge devices. This leads to heavy network traffic and makes real-time network traffic prediction more difficult.
2) ML-based network traffic prediction management algorithms require a large number of training dataset collected from the prior network traffic of edge devices. Storing the network traffic data needs huge memory space and consumes many computing resources for network traffic management in the server side.
3) On the contrary, for network edge devices, it indeed consumes much computing and memory resources to collect prior network traffic as training dataset. Moreover, some edge devices cannot provide sufficient memory

resources for collecting enough network traffic data. In addition, it is difficult for edge devices to train a network traffic prediction algorithm by using traditional training approaches.

4) In general, the dataset generated from IIoT devices has heterogeneous data ranges. That is, the ranges of time series generated from different devices vary greatly. Thus, the cloud-side prediction approach has to normalize the raw data to a uniform range and then constructs a universal model to predict the network traffic values in a normalized sense, and then outputs the de-normalized prediction value for each edge device. However, the final output prediction error might be large when the algorithm converts prediction values in the uniform range back to the original range during testing.

Thus, in IIoT, there is a dilemma in choosing either to utilize centralized network traffic prediction in the server side or to employ network traffic measurement on edge side. Although several previous works such as [4] pointed out that it seems to be infeasible to train a network traffic prediction in edge devices, but it is indeed a possible way to lessen the workload of the management server.

Motivated by the aforementioned issues, we focus on the problem of IIoT edge network traffic prediction. Deploying deep learning models into IIoT edge devices is facing several difficult challenges. At first, to guarantee that inference efficiency in an IIoT edge device, the chosen prediction models should be lightweight which may cause underfitting. On the other hand, since the memory resource of the edge device is limited, it only allows the edge devices to keep few prior sampling data in memory as training dataset. However, this will result in overfitting when the number of training epochs is large. Furthermore, the number of training epochs in the edge training should be strictly small so that the edge devices have short training duration and preserve real-time execution. Thus, constructing an edge network traffic prediction algorithm with fast adaptation when having few data is the goal of this article. To achieve it, we propose a meta-learning-based mechanism for real-time edge traffic prediction. The main idea is to learn how to adapt model parameters quickly by looking at many diverse traffic prediction tasks from different edge devices. We use the meta-learning-based approach to realize it. In meta learning, a lot of prediction tasks are required. Fortunately, it is easy for us to collect many traffic prediction tasks generated from lots of IIoT edge devices of the same type. Then, we extend model-agnostic meta-learning (MAML) [5] to edge network traffic prediction. IIoT edge devices of the same type are divided into two groups training devices and test devices. We define two subsets called support data and query data which consist of traffic time series generated from training devices. After training by support and query data in a reinforcement learning way, a good initial weight of the lightweight prediction model is learned. Then, starting from this learned initial weight, the lightweight model of the test device can quickly adapt to a suitable model weight via its short traffic time series. This fulfills the goal of this article. We summarize main contributions of our work as follows.

1) An edge network traffic prediction framework is proposed in order to ease the workload of IIoT cloud server. To ensure the edge-side prediction efficiency, we also suggest several lightweight models which can be easily deployed in the edge devices.
2) This article shows a meta-learning-based method to train lightweight traffic prediction models quickly within edge devices. Initially, the proposed method makes the cloud server to generate good initial model weights for lightweight models deployed in edge devices. Then, starting from the model weight generated by the server, the lightweight model parameters can be adapted quickly to each edge traffic prediction algorithm.
3) To realize the proposed meta-learning-based method, edge devices of the same type are divided into two parts training devices and test devices. Two datasets, support data and query data, are generated from training devices. Based on two datasets, we define the meta learning problem as how to adapt fast to a new network traffic time series generated from a test edge device given a set of known network traffic time series in support and query data. To address this meta learning problem, the article is the first work to extend the model agnostic meta-learning (MAML) method [5] to network traffic prediction.
4) By conducting experiments on real datasets, this article shows the feasibility of network traffic prediction realized in the edge side. Furthermore, we also demonstrate the advantage of edge-side network traffic prediction via a prediction performance comparison with the cloud-side traffic prediction algorithm previously developed by Wang et al. [6]. In fact, experimental results show that our edge-side MAML-based prediction method outperforms the cloud-side prediction method in [6] since our approach utilizes the good property of the homogeneous data range.

The organization of the rest paper is as follows: In Section II, some related works are reviewed. Section III gives some background knowledge of MAML. In Section IV, we give the detailed problem description of this paper and propose our meta-learning-based edge network traffic prediction method. In Section V, the experimental results are given. Finally, a conclusion is given in Section VI.

II. Related Work

A. Machine-Learning-Based Network Traffic Prediction

Recently, machine learning has been used in network traffic prediction to extract features of the network traffic time series. In [7], Alvizxu et al. constructed a feedforward neural network with three layers to predict network traffic in a mobile metro-core network based on a software defined network. In [8], Qiu et al. studied the spatio-temporal features of the wireless network traffic where several recurrent neural networks were constructed for the network traffic prediction. In [9], Singh and Jukan utilized machine learning for prediction of time-varying traffic and connection blocking in optical data center networks. In [10], Zhao et al. proposed a deep LSTM-

based recurrent neural network to extract the spatiotemporal features for network traffic prediction in large-scale backbone networks. In [11], Zhang et al. proposed a spatialtemporal cross-domain neural Network to effectively capture the data patterns hidden in cellular network. They also introduced an inter-cluster transfer learning strategy to efficiently reuse the extracted knowledge. In [12], Zhang et al. used a deep learning method to model the dynamics of wireless network traffic. By treating traffic data as images, they constructed a convolutional neural network (CNN) to predict network traffic in cellular network where the constructed CNN can learn the spatial and temporal dependence. As mentioned in [13], these methods are not suitable to be deployed in IIoT due to dense infrastructure deployment and high energy consumption.

B. Cloud-side Network Traffic Prediction in IIoT

Machine learning technology greatly helps IIoT to improve production efficiency and communication intelligence. However, as the IIoT traffic data increases rapidly, we have to trade the spectrum resources against network energy consumption. Network traffic prediction is one of methods to address this issue for achieving intelligent IIoT. In [14], Sun et al. proposed an integrated neural computing model called enhanced echo-state restricted Boltzmann machine to predict network traffic. In [15], Lu and Yang used long short-term memory (LSTM) neural network to construct a real-time network traffic prediction model by modifying the loss function of LSTM network. In [16], Ko et al. transformed the network traffic prediction into a classification problem and utilized CNN technique to classify the fixed time interval traffic flow. In [17], Zhang et al. proposed an LSTM-based network traffic prediction model to forecast network traffic timely and accurately. In [13], Nie et al. proposed a reinforcement learning-based prediction framework by modeling the network traffic prediction problem as a Markov decision process. Then the network traffic is predicted via Monte Carlo Q-learning. In [18], Nie et al. observed that the large number of prior network traffic information needs to consume expensive network resources for sampling. Then they designed a multitask-learning-based prediction algorithm and used link load as extra information to increase the prediction accuracy. In [6], Wang et al. designed an algorithm called Flow2graph to predict network traffic in IIoT. Their algorithm extracts the so-called shapelets from the network traffic sequence and converts the traffic flow into a shapelets conversion graph based on the relationship between the traffic sequence and shapelets. Then, a graph isomorphism network is utilized to learn the specificity of the traffic flow from different devices and predict the future traffic values. By above discussion, a lot of works utilized deep learning techniques to predict IIoT network traffic. However, most works focus on predicting traffic of backbone networks and lack of predicting network traffic of specific devices. Although Wang et al.'s work [6] considered the specificity of IIoT devices, their proposed method is still not suitable to be deployed in edge devices since the learned shaplets may not be applied to the traffic flow generated from edge devices of new type. Consequently, it costs much effort for the cloud server to update the set of learned shaplets for these new edge devices and results in large cloud-side workload. In addition, both the model size and the inference time of their proposed algorithm are too still large to be deployed in edge side. Therefore, this article proposed an edge-side network traffic prediction model with small model size, short inference time, and easy adaptation to new edge devices.

C. Meta-Learning in Time Series Prediction

Deep learning has demonstrated great performance on many tasks in different areas. However, deep neural networks usually fail to learn new tasks well with limited amounts of data. Meta learning is one of solutions to overcome this limitation. The basic idea is to learn an effective learning algorithm over a large number of different tasks such that the target tasks can be learned from a few data examples. In [5], Finn et al. proposed a model-agnostic meta learning algorithm (MAML) where the model parameters are explicitly trained such that only a few of gradient steps with a small amount of training examples from a new task are required to have good generalization performance on that task. In [19], the MAML method is extended to time series regression. The authors observed that, in most time series regression problems, it is usual to have very long but a small number of time series. Thus, they proposed a partition method for generating diverse tasks by assuming that few but long time series are at hand. However, in IIoT, the network traffic time series generated by edge devices are usually short due to limited memory resource. Thus, the MAML-based time-series forecasting approach suggested in [19] is inapplicable in the scenario of IIoT network traffic prediction. This article proposes the first MAML-based framework for IIoT edge network traffic prediction.

III. Preliminaries

A. Model-Agnostic Meta-Learning

Model-Agnostic Meta-Learning (MAML) [5] is a meta-learning algorithm which can solve different tasks such as simple regression, image classification, reinforcement learning, etc. In this section, we briefly review the MAML algorithm.

In the MAML setting, let $\{T_1, T_2, \ldots, T_n\}$ denote the batch of tasks sampling from a distribution over tasks $p(T)$ where $T_i = (D_i^{supp}, D_i^{qry})$ for the support data D_i^{supp} and the query data D_i^{qry} of the task T_i. In addition, let L_{T_i} denote the loss function of the task T_i and let $L_{T_i}(f_\theta, D)$ denote the loss value with respect to the dataset D and a neural network f_θ with parameter θ.

The meta-training step of MAML is described as follows. In the beginning of the training process, the initial parameter θ is randomly initialized. During meta-training for the i-th task T_i, the model f_θ is trained by using the dataset D_i^{supp} and the parameter θ is updated to θ_i'. Precisely, for the task T_i, we define θ_i' by

$$\theta_i' \triangleq \theta - \alpha \nabla_\theta L_{T_i}(f_\theta, D_i^{supp}). \tag{1}$$

where α is a fixed learning rate. Next, we evaluate the loss of $f_{\theta_i'}$ by the query dataset D_i^{qry}, that is $L_{T_i}(f_{\theta_i'}, D_i^{qry})$. Then we

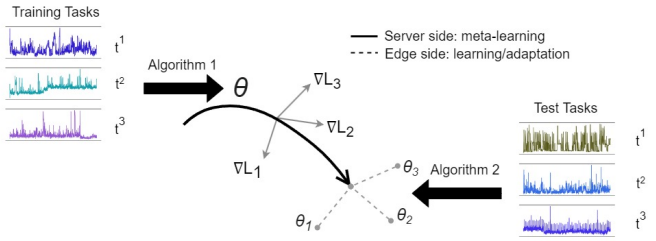


Fig. 1: An overview of our IIoT network traffic prediction framework. The framework consists of two parts: a cloud-side prediction model weight initialization procedure (Algorithm 1) and an edge-side model adaption and prediction procedure (Algorithm 2). Algorithm 1 finds a good initial parameter vector θ. For a specific edge device E, starting from θ, Algorithm 2 quickly finetunes the lightweight traffic prediction model to a parameter vector θ_E.

expect to compute the argument of the following minimization.

$$\tilde{\theta} \triangleq \arg\min_{\theta} \sum_{i=1}^{n} L_{T_i}(f_{\theta'_i}, D_i^{qry}). \tag{2}$$

Instead of computing $\tilde{\theta}$, we compute the gradient decent for all datasets D_i^{qry} and use it to update the parameter θ as follows.

$$\theta \longleftarrow \theta - \beta \nabla_{\theta} \sum_{i=1}^{n} L_{T_i}(f_{\theta'_i}, D_i^{qry}). \tag{3}$$

where β is the fixed meta-learning rate. The meta-training step is repeated for each training task T_i until convergence.

After completing meta-training, the meta-testing is carried out as follows. First, a new task $T = (D^{supp}, D^{qry})$ and its loss function L_T are given. Next, we compute

$$\phi \longleftarrow \theta - \alpha \nabla_{\theta} L_T(f_{\theta}, D^{supp}). \tag{4}$$

Finally, we make predictions on D^{qry} by using the weighted neural network f_{ϕ}.

IV. Meta-Learning-Based Edge Network Traffic Prediction

In this section, we will give a detailed description of our MAML-based edge network traffic prediction framework. The proposed framework consists of a cloud-side prediction model weight initialization procedure (Algorithm 1) and an edge-side model adaption and prediction procedure (Algorithm 2). An overview of our proposed method is shown in Fig. 1.

A. Prediction Model Weight Initialization

Instead of training edge network traffic prediction model from scratch, we will find a good initial model weight such that, starting from this learned initial weight, a new edge-side prediction model can quickly adapt to a nice model weight by using its short traffic time series.

Here, we introduce how the cloud server determines a good initial model weight of the edge-side prediction model based on MAML. First of all, we define the meta learning for IIoT network traffic prediction as follows. Given a set of edge devices $E_1, \ldots, E_N$ of the same type, let $\mathbf{t}^1, \mathbf{t}^2, \ldots, \mathbf{t}^N$ be the time series collected from edge devices $E_1, E_2, \ldots, E_N$, respectively. The length of each time series $\mathbf{t}^i$ is assumed to be short due to the limited memory resource of edge devices and the cloud server. We remark that the number N will be large enough since there are usually many edge devices of the same type deployed in an IIoT system. For the time series $\mathbf{t}^i$, we define $D_i = \{(\mathbf{x}_n, y_n)\}_{n=1:K}$ where $\mathbf{x}_n \in \mathbb{R}^{w \times \ell}$ is a time window from $\mathbf{t}^i$ which is of width w and has label $y_n \in \mathbb{R}$. Specifically, in our IIoT setting, we assume that $w = 1$, $\mathbf{x}_n = \mathbf{t}^i_{[(n-1)s+1:(n-1)s+\ell]}$, and $y_n = \mathbf{t}^i_{(n-1)s+\ell+1}$ where s is the stride size. Next, D_i is partitioned into two parts $D_i^{supp} = \{(\mathbf{x}_n, y_n)\}_{n=1:Q}$ and $D_i^{qry} = \{(\mathbf{x}_n, y_n)\}_{n=Q+1:K}$. Now we define the i-th training task by

$$T_i \triangleq (D_i^{supp}, D_i^{qry}). \tag{5}$$

Since each task T_i is a regression problem, we define the loss function $L_{T_i} = L_{mse}$ where L_{mse} is the mean squared error (MSE). In addition, we use f_{θ} to denote the lightweight neural network with the parameter θ. In Section V, we will suggest several lightweight models as candidates for f. Based on the above system setting, the goal of the cloud server is to find a good initial model weight θ by using the MAML algorithm.

The algorithm is shown in Algorithm 1. At first, we initialize

Algorithm 1 Cloud-side Edge Model Initialization Procedure

Require: f: a given neural network
Require: $\mathbf{T} = \{T_1, \ldots, T_N\}$: training tasks
Require: α, β: learning rates
1: Initialize a random θ
2: **while** not done **do**
3: Randomly sample a batch of tasks $\mathbf{B}$ from $\mathbf{T}$
4: **for all** $T_i \in \mathbf{B}$ **do**
5: $\theta_i \longleftarrow \theta$
6: **for** $j \leftarrow 1$ to Q **do**
7: $\theta_i \longleftarrow \theta_i - \alpha \nabla_{\theta} L_{mse}(f_{\theta}(\mathbf{x}_j), y_j)$
8: **end for**
9: **end for**
10: Update $\theta = \theta - \beta \nabla_{\theta} \sum_{i=1}^{|\mathbf{B}|} \sum_{j=Q+1}^{K} L_{mse}(f_{\theta_i}(\mathbf{x}_j), y_j)$
11: **end while**

random model parameter vector θ for the neural network f_{θ}. In each iteration, we randomly sample a fixed number of tasks $\mathbf{B}$ from the set of training tasks $\mathbf{T}$. For each sampled task $T_i \triangleq (D_i^{supp}, D_i^{qry})$, we compute the parameter vector θ_i by sequentially applying gradient decent to f_{θ} for the first Q examples $(\mathbf{x}_j, y_j) \in D_i^{supp}$. After computing all θ_i, we compute the gradient $\nabla_{\theta} \sum_{i=1}^{N} \sum_{j=Q+1}^{K} L_{mse}(f_{\theta_i}(\mathbf{x}_j), y_j)$ and use it to update the original parameter vector θ by $\theta - \beta \nabla_{\theta} \sum_{i=1}^{N} \sum_{j=Q+1}^{K} L_{mse}(f_{\theta_i}(\mathbf{x}_j), y_j)$. After completing the training, the learned model parameter vector will be sent to the corresponding edge device for efficient adaptation to its network traffic prediction task which will be described in the next section.

B. Adapting to Prediction Tasks of IIoT Edge Devices

As the previous discussion, we assume that the edge device E has received a good initial parameter vector θ_E computed by the cloud server for the lightweight neural network f_{θ_E}.

For the time series $\mathbf{t}$ collected by the edge device E, we define $D = \{(\mathbf{x}_n, y_n)\}_{n=1:K}$ where $\mathbf{x}_n = \mathbf{t}_{[(n-1)s+1:(n-1)s+\ell]}$, and $y_n = \mathbf{t}_{(n-1)s+\ell+1}$ by using s as the stride size. Then, D is partitioned into two parts $D^{supp} = \{(\mathbf{x}_n, y_n)\}_{n=1:Q}$ and $D^{qry} = \{(\mathbf{x}_n, y_n)\}_{n=Q+1:K}$. Let T be the new task defined by $T \triangleq (D^{supp}, D^{qry})$ and let the loss function L_T be L_{mse}. Based on the above edge system setting, the goal of the edge device is to make the neural network f_{θ_E} quickly adapt to the task T.

The edge-side adaptation algorithm is shown in Algorithm 2. In the beginning, the adaptation algorithm only fine-tunes f within a few epochs starting from the initial parameter vector θ_E by using the dataset D^{supp}. After fine-tuning, the algorithm outputs f_{θ_E} to predict on the dataset D^{qry}.

Algorithm 2 Edge-side Adaptation Procedure

Require: f_{θ_E}: the lightweight model with initial parameter vector θ_E deployed in the edge device E
Require: $T \triangleq (D^{supp}, D^{qry})$: a new task generated by the device E
Require: α: adaptation learning rate
Require: e: the number of epochs
for $z = 1$ to e **do**
  for all $(\mathbf{x}_j, y_j) \in D^{supp}$ **do**
    $\theta_E \longleftarrow \theta_E - \alpha \nabla_\theta L_{mse}(f_\theta(\mathbf{x}_j), y_j)$
  end for
end for
Output f_{θ_E} to predict on D^{qry}

V. Experiments

In this section, we conduct several experiments for our IIoT network traffic prediction method and baseline methods on several different time series datasets. All experimental algorithms are implemented on NVIDIA RTX 4090 GPU and implemented in PyTorch [20].

A. Datasets

We evaluate the performance of our models on three different time series datasets described as follows.

FedCSIS 2020 Challenge [6], [21] (Denoted as **F-dataset**) is a dataset which collected workload data extracted from 24249 IT devices over 80 days. It contains hourly aggregated values of various workload characteristics such as "cpu_usage", "memoory_free" extracted from device logs. In this article, we select 5 kinds of workload characteristics ("cpu_usage", "memoory_free", "memoryallocatedbyproc", "in_traffic", and "out_traffic") as our experimental sub-datasets and focus on predicting "Mean" (the mean of the values) or "Volume" (the number of values) in our experiments.

Urban Traffic Speed Dataset of Guangzhou [22] (Denoted as **G-dataset**) contains traffic speed data of 214 road segments within two months (61 days from August 1, 2016 to September 30, 2016) with a 10-minute interval (144 intervals per day) in Guangzhou, China. In our experiments, we take the first 2000 data points of each time series in the original dataset as our experimental dataset.

Seattle Inductive Loop Detector Dataset [22]–[24] (Denoted as **S-dataset**) collects freeway traffic speed in Seattle over the whole year of 2015. The time interval of the data is 5-minute. The data is collected from 323 loop detectors. We choose the data from January 1 to January 28 as our experimental dataset.

In particular, the sizes of time window for **F-dataset**, **G-dataset**, and **S-dataset** are 168, 144, and 144, respectively. The stride of these time window is 1 for all dataset. The ratio of the support set size $|D^{supp}|$ to the query set size $|D^{qry}|$ for **F-dataset**, **G-dataset**, and **S-dataset** are all 8:2.

To conduct our experiment for our meta-learning approach, for time series in each dataset, we divide them into two groups: one is called the set of training tasks and the other is called the set of test tasks. The ratio of the number of training tasks to the number of test tasks is 8:2.

B. Implementation Details

Our MAML algorithm is implemented by utilizing `algorithm` module from `learn2learn` library [25]. Note that the network traffic prediction algorithms are required to be deployed in edge devices. Thus, these prediction algorithms should be lightweight. Here, we consider three lightweight models 1-dimensional convolutional neural network (CNN), self-attention (Self-Att) and CNN-LSTM as candidates. For **F-dataset**, architectures of three models are shown in Table I, Table II, and Table III, respectively. In addition, for **G-dataset** and **S-dataset**, architectures of three models are shown in Table IV, Table V, and Table VI, respectively. (BR) indicates batch normalization and ReLU. The stride is 1 and Zero padding is applied for each convolution layer. For Max Pooling, the stride is 2.

TABLE I: 1-D CNN architecture for **F-dataset**

Type/Stride	Filter Shape	Input Size
Conv BR/s1	5 x 1 x 64	168 x 1
Max Pool/s2	Pool 2	168 x 64
$Depth_1$, Tensor shape: 84 x 64		
Conv BR/s1	3 x 64 x 128	84 x 64
Max Pool/s2	Pool 2	84 x 128
$Depth_2$, Tensor shape: 42 x 128		
Conv BR/s1	3 x 128 x 256	42 x 128
Max Pool/s2	Pool 2	42 x 256
$Depth_3$, Tensor shape: 21 x 256		
Flatten, Tensor shape: 5376		
FC layer1	512 x 5376	5376
FC layer2	64 x 512	512
FC layer3	1 x 64	64

In Table VII, the model size and the parameter size of three lightweight models and Flow2graph model [6] are shown. As one can see, Self-Att model is the most lightweight among these models.

In our experiments, all lightweight models are trained with the same hyperparameters for all support datasets D^{supp}

TABLE II: CNN-LSTM architecture for **F-dataset**

Type/Stride	Filter Shape	Input Size
Conv BR/s1	5 x 1 x 64	168 x 1
Max Pool/s2	Pool 2	168 x 64
$Depth_1$, Tensor shape: 84 x 64		
Conv BR/s2	3 x 64 x 128	84 x 64
Max Pool/s2	Pool 2	84 x 128
$Depth_2$, Tensor shape: 42 x 128		
Conv BR/s2	3 x 128 x 256	42 x 128
Max Pool/s2	Pool 2	42 x 256
$Depth_3$, Tensor shape: 21 x 256		
LSTM	[256 x 256] x 2	21 x 256
Take last time step, Tensor shape: 256		
FC layer	1 x 256	256

TABLE III: Self-Att architecture for **F-dataset**

Type/Stride	Filter Shape	Input Size
FC layer	64 x 1	168 x 1
attention	64 x 64	168 x 64
Take last time step, Tensor shape: 64		
FC layer1	64 x 64	64
FC layer2	1 x 64	64

and query datasets D^{qry} generated from mentioned datasets. During training on D^{supp} and D^{qry}, the optimizer is Adam and the learning rate, the weight decay, and the number of training epochs are 0.0005, 0.0001, and 30, respectively. The batch size is to 128 and 32 for D^{supp} and D^{qry}.

Before the meta-learning step, we set hyperparameters $\alpha = 0.001$ and $\beta = 0.0005$. The number of sampling tasks (i.e. $|\mathbf{B}|$) in **F-dataset**, **G-dataset** and **S-dataset** are 128, 64, and 64, respectively.

We also carry out data preprocessing for mentioned datasets. For each time series in these three datasets, we use standardization to normalize each of them. To adjust the outliers in each time series in **F-dataset**, we use well-known *3-sigma* method to process it. First, we calculate the mean and standard deviation of each time series. Then, we determine the upper and lower bounds by multiplying the standard deviation by three and adding or subtracting it from the mean. Any data that fall outside of this range are regarded as outliers. Finally, we replace the outliers with upper or lower bounds.

Remark that this article proposes the first edge-side network traffic prediction framework based on MAML. To compare the proposed prediction models (i.e. these lightweight models with initial parameter vectors generated by Algorithm 1), we defined three mentioned lightweight models with random initial parameter vectors as baseline models. For in-task training, baseline models are trained from for scratch in 30 epochs. The learning setting and hyperparameters are the same as the setting of the proposed models. During the fine-tuning, their performance will be compared every epoch.

C. Metrics

To assess the performance in time series prediction, we use mean absolute percentage error (MAPE), root mean squared error (RMSE), and mean absolute error (MAE) as performance metrics.

TABLE IV: 1-D CNN architecture for **G-dataset** and **S-dataset**

Type/Stride	Filter Shape	Input Size
Conv BR/s1	5 x 1 x 8	144 x 1
Conv BR/s1	5 x 8 x 16	140 x 8
Conv BR/s1	3 x 16 x 32	136 x 16
Flatten, Tensor shape: 8448		
FC layer1	64 x 8448	8448
FC layer2	1 x 64	64

TABLE V: CNN-LSTM architecture for **G-dataset** and **S-dataset**

Type/Stride	Filter Shape	Input Size
Conv BR/s1	5 x 1 x 8	144 x 1
Max Pool/s2	Pool 2	144 x 8
$Depth_1$, Tensor shape: 72 x 8		
Conv BR/s2	3 x 8 x 16	72 x 8
Max Pool/s2	Pool 2	72 x 16
$Depth_2$, Tensor shape: 36 x 16		
Conv BR/s2	3 x 16 x 32	36 x 16
Max Pool/s2	Pool 2	36 x 32
$Depth_3$, Tensor shape: 18 x 32		
LSTM	[32 x 32] x 2	18 x 32
Take last time step, Tensor shape: 32		
FC layer	1 x 32	32

D. Comparison Results

In this section, we compare the performance of the proposed framework and baseline models. For **F-dataset**, we only show results when the models predict the value "Mean" of **F-dataset** and put the results in the appendix when the models predict the value "Volume" of **F-dataset**. First of all, we compare the prediction performance when the number of epochs for adaptation is 5. The comparison result is shown in Fig. 2 where it shows that the performance of our MAML-based model outperforms that of the baseline model in most cases. This gives evidence that our proposed IIoT edge prediction model can quickly adapt to the traffic time series generated by the corresponding edge device.

In addition, due to its small model size and small number of parameters, we suggest to use Self-Att model as the lightweight traffic prediction model deployed in edge devices. In Fig. 3, we show the loss curves generated by our proposed model and the baseline model for the Self-Att model. In these figures, the green point and the red point stand for the evaluation performance of the proposed model and the baseline model, respectively. As shown in Fig. 3, the quick adaptation of the proposed MAML-based Self-Att model is guaranteed. Thus, based on small model size and quick adatation, it greatly ensures the edge devices to carry out the fine-tuning process in the edge size.

To sum up, for the real-time edge-side network traffic training and prediction, our proposed MAML-based lightweight models clearly outperform the baseline models. In particular, we strongly suggest to use the Self-Att model as the edge-side traffic prediction model because of its small model size which is applicable for edge devices with limited memory resource.

TABLE VI: Self-Att architecture for **G-dataset** and **S-dataset**

Type/Stride	Filter Shape	Input Size
FC layer	8 x 1	144 x 1
attention	8 x 8	144 x 8
Take last time step, Tensor shape: 8		
FC layer1	8 x 8	8
FC layer2	1 x 8	8

TABLE VII: Comparison of parameter size and model size among three lightweight models and Flow2graph model [6]

Model	Dataset	Size	Parameters
CNN	**F-dataset**	11.64 (MB)	2,910 K
Self-Att	**F-dataset**	0.08 (MB)	20 K
CNN-LSTM	**F-dataset**	6.81 (MB)	1,703 K
Flow2graph [6]	**F-dataset**	16.92 (MB)	5,016 K
CNN	**G-dataset** & **S-dataset**	2.2 (MB)	549 K
Self-Att	**G-dataset** & **S-dataset**	1.54 (KB)	0.385 K
CNN-LSTM	**G-dataset** & **S-dataset**	0.11 (MB)	27 K

E. Ablation Study I: Edge-based versus cloud-based IIoT network traffic prediction

In this ablation study, we compare the prediction performance between our proposed edge-side IIoT network traffic prediction and the cloud-side prediction algorithm of Wang et al. [6] which is called Flow2graph. Here we modify the output of Flow2graph in order to fit our performance evaluation setting. Precisely, the original output length of Flow2graph is 24 and we only choose its first output digit as the prediction output and discard the rest. The detail of our experiment is as follows. Firstly, we train Flow2graph according to the training setting used in [6]. Secondly, we also use five sub-datasets F-cpu_usage, F-memoory_free, F-memoryallocatedbyproc, F-in_traffic, and F-out_traffic to construct five testing sub-datasets by excluding their time series which are used in the training step of Flow2graph from the above sub-datasets in order to make sure that the test data is unseen in the training step. For example, if the number of devices (time series) collected in the sub-dataset "cpu_usage" is 493 and there are 190 time series used for training Flow2graph, then the remaining 303 time series are used to construct testing sub-dataset. Then we evaluate the trained Flow2graph by using the five testing sub-datasets. The comparison result is shown in Fig. 4 where it shows that our MAML-based Self-Att model greatly outperforms Flow2graph in most cases under the metrics RMSE, MAPE and MAE. This demonstrates the advantage of edge-side IIoT network traffic prediction.

F. Ablation Study II: Performance comparison in different length of time series used for training dataset construction

In this ablation study, we discuss why we only use first 2000 data points of the time series in all datasets. In Table VIII, the MAE performance results of different sampling lengths are shown. As one can see, two Self-Att models with sampling length 2000 and 2500 have closed MAE performance. In order to save the memory resource of IIoT devices, we then suggest

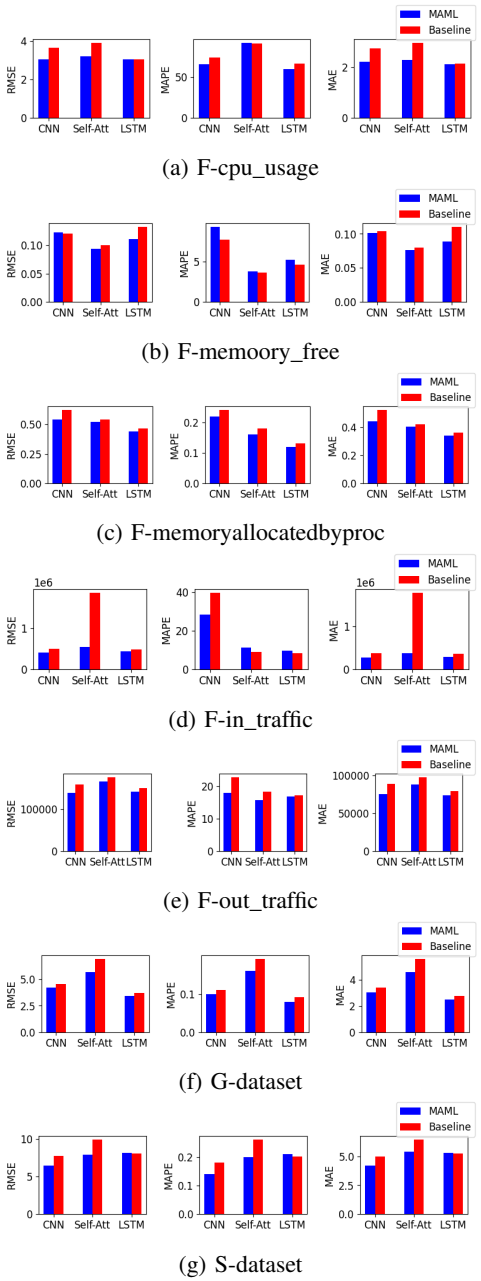
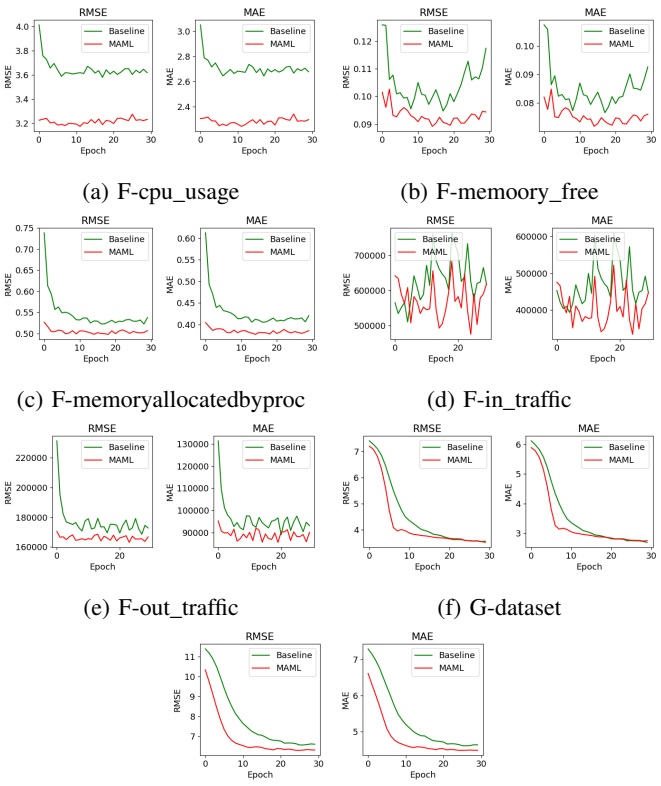


Fig. 2: Performance comparison with MAPE, RMSE, and MAE between our MAML-based model and baseline model on **F-dataset**, **G-dataset** and **S-dataset**.

to use the prediction model with sampling length 2000 in the case that the prediction model is Self-Att.

VI. Conclusion

In this paper, we study the problem of IIoT network traffic prediction. Previous works mainly focused on constructing traffic prediction models for the IIoT cloud-side server. Their approach may result in huge computing and memory consumption for the cloud server and lack of device specificity. To address this problem, instead of cloud-side approach, we propose an edge-side IIoT network traffic prediction framework which consists of a cloud-side initial model parameter



(a) F-cpu_usage (b) F-memoory_free

(c) F-memoryallocatedbyproc (d) F-in_traffic

(e) F-out_traffic (f) G-dataset

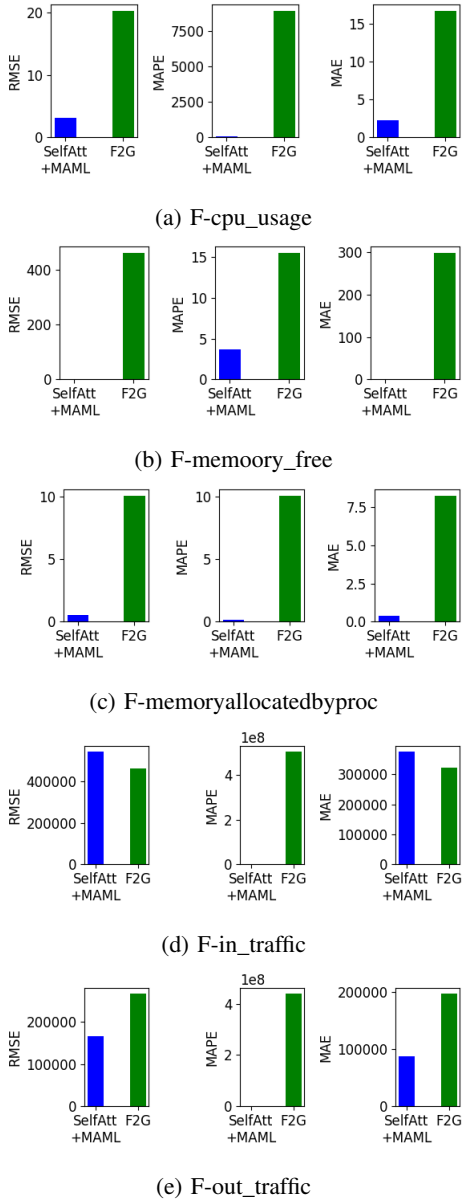
(g) S-dataset

Fig. 3: Performance comparison of Self-Att model with RMSE and MAE between our MAML-based model and the baseline model on **F-dataset**, **G-dataset**, and **S-dataset**.

TABLE VIII: MAE performance in 5 epochs and 10 epochs for our MAML-based model and the baseline model on **G-dataset** under different sampling length settings. The experimental model is Self-Att.

Length	5-epoch		10-epoch	
	MAML	Baseline	MAML	Baseline
1500	5.45	5.62	3.42	4.34
2000	4.59	5.58	3.13	3.77
2500	4.55	5.16	3.27	3.29

vector generating algorithm and an edge-side fine-tuning and prediction algorithm. By utilizing the MAML technique which is one of meta-learning methods, the cloud server is able to generate a good initial parameter vector for the lightweight models deployed in edge devices. Then, starting with this good initial parameter vector, the edge-side learning algorithm has quick adaptation to fulfill the traffic prediction task of the corresponding edge device. Moreover, we also suggest several lightweight models to be deployed in edge devices for traffic prediction. Via a thorough experimental analysis, the Self-Att model is recommended to be a good lightweight model candidate which has a very small number of parameters. Finally, experimental results show that our MAML-based framework greatly outperforms not only the train-from-scratch approach but also the state-of-the-art cloud-side prediction method.



(a) F-cpu_usage

(b) F-memoory_free

(c) F-memoryallocatedbyproc

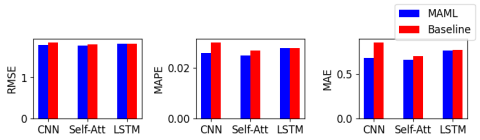
(d) F-in_traffic

(e) F-out_traffic

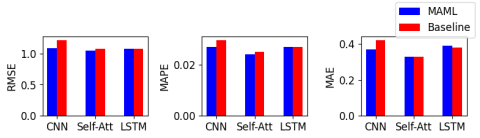
Fig. 4: Performance comparison with MAPE, RMSE, and MAE between our edge-side MAML-based Self-Att model and the cloud-side traffic prediction model of Wang et al. [6] (denoted as F2G in figures) on five testing sub-datasets of **F-dataset**.

APPENDIX

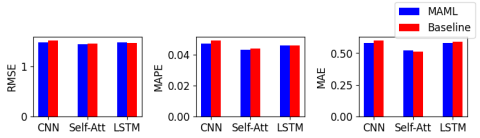
In the appendix, we show more comparison results between the proposed MAML-based model and baseline model in the case that models predict the value "Volume" in **F-dataset**. Similar to Section V-D, we compare the prediction performance when the number of epochs for adaptation is 5. The comparison result is shown in Fig. 5 where it shows that the performance of our MAML-based model outperforms that of the baseline model in most cases. It demonstrates that our proposed IIoT edge prediction model can quickly adapt to the traffic time series generated by the corresponding edge device. In Fig. 6, we also show the loss curves generated by



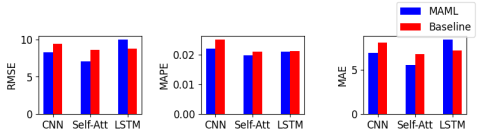
(a) F-cpu_usage



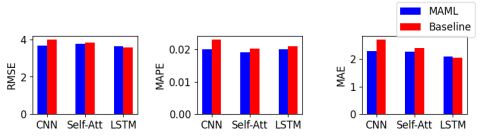
(b) F-memoory_free



(c) F-memoryallocatedbyproc

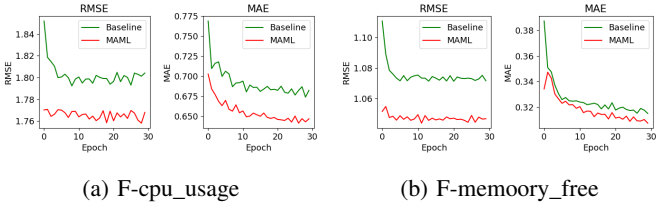


(d) F-in_traffic



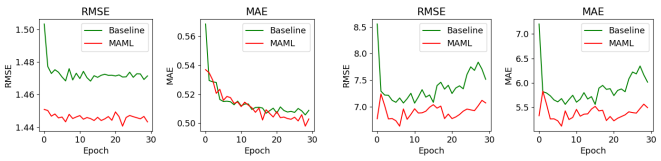
(e) F-out_traffic

Fig. 5: Performance comparison with MAPE, RMSE, and MAE between our MAML-based model and baseline model on **F-dataset** when models predict the value "Volume".



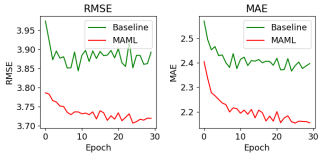
(a) F-cpu_usage

(b) F-memoory_free



(c) F-memoryallocatedbyproc

(d) F-in_traffic



(e) F-out_traffic

Fig. 6: Performance comparison of Self-Att model with RMSE and MAE between our MAML-based model and the baseline model on **F-dataset** when models predict the value "Volume".

our proposed model and the baseline model for the Self-Att model.

REFERENCES

[1] F. Khan, M. A. Jan, A. ur Rehmnan, S. Mastorakis, M. Alazab, and P. Watters, "A secured and intelligent communication scheme for IIOT enabled pervasive edge computing," *IEEE Tansactions on Industrial Informatics*, vol. 17, no. 7, pp. 5128–5137, 2021.

[2] Y. Xu, F. Yin, W. Xu, J. Lin and S. Cui,"Wireless traffic prediction with scalable Gaussian process: Framework algorithms and verification," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1291–1306, Jun. 2019.

[3] L. Nie, X. Wang, S. Wang, Z. Ning, M. Obaidat, B. Sadoun, and S. Li, "Network traffic prediction in industrial Internet of Things backbone networks: A multitask learning mechanism," *IEEE Trans. Ind. Informat.*, vol. 17, no. 10, pp. 7123-7132, Oct. 2021.

[4] L. Nie, Z. Ning, M. Obaidat, B. Sadoun, H. Wang, S. Li, L. Guo, and G. Wang, "A Reinforcement Learning-Based Network Traffic Prediction Mechanism in Intelligent Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 17, no. 3, pp. 2169-2180, March. 2021.

[5] C. Finn, P. Abbeel and S. Levine,"Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. Int. Conf. Mach. Learn.*, pp. 1126–1135, 2017.

[6] R. Wang, Y. Zhang, L. Peng, G. Fortino and P.-H. Ho, "Time-Varying-Aware Network Traffic Prediction Via Deep Learning in IIoT," *IEEE Transactions on Industrial Informatics,* vol. 18, no. 11, pp.8129–8137, Nov. 2022.

[7] R. Alvizu, S. Troia, G. Maier and A. Pattavina, "Matheuristic with machine-learning-based prediction for software-defined mobile metro-core networks," *Journal of Optical Communications and Networking,* vol. 9, no. 9, pp.D19–D30, Sept. 2017.

[8] C. Qiu, Y. Zhang, Z. Feng, P. Zhang and S. Cui, "Spatio-temporal wireless traffic prediction with recurrent neural network," *IEEE Wireless Commun. Lett.,* vol. 7, no. 4, pp.554–557, Aug. 2018.

[9] S. K. Singh and A. Jukan, "Machine-learning-based prediction for resource (re)allocation in optical data center networks," *IEEE/OSA J. Opt. Commun. Netw.,* vol. 10, no. 10, pp.D12–D28, Oct. 2018.

[10] J. Zhao, H. Qu, J. Zhao and D. Jiang, "Towards traffic matrix prediction with LSTM recurrent neural networks," *Electron. Lett.,* vol. 54, no. 9, pp.566–568, 2018.

[11] C. Zhang, H. Zhang, J. Qiao, D. Yuan and M. Zhang, "Deep transfer learning for intelligent cellular traffic prediction based on cross-domain big data," *IEEE J. Sel. Areas Commun.,* vol. 37, no. 6, pp.1389–1401, Jun. 2019.

[12] C. Zhang, H. Zhang, D. Yuan and M. Zhang, "Citywide cellular traffic prediction based on densely connected convolutional neural networks," *IEEE Commun. Lett.,* vol. 22, no. 8, pp.1656–1659, Aug. 2018.

[13] L. Nie et al., "Network Traffic Prediction in Industrial Internet of Things Backbone Networks: A Multitask Learning Mechanism," *IEEE Transactions on Industrial Informatics,* vol. 17, no. 10, pp.7123–7132, Oct. 2021.

[14] X. Sun et al., "Enhanced echo-state restricted Boltzmann machines for network traffic prediction," *IEEE Internet Things J.,* vol. 7, no. 2, pp.1287–1297, Feb. 2020.

[15] H. Lu and F. Yang, "Research on network traffic prediction based on long short-term memory neural network," in *Proc. IEEE 4th Int. Conf. Comput. Commun.,* pp.1109–1113, 2018.

[16] T. Ko, S. M. Raza, D. T. Binh, M. Kim and H. Choo, "Network prediction with traffic gradient classification using convolutional neural networks," in *Proc. IEEE 14th Int. Conf. Ubiquitous Inf. Manage. Commun.,* pp.1–4, 2020.

[17] L. Zhang et al., "LNTP: An end-to-end online prediction model for network traffic," *IEEE Netw.,* vol. 35, no. 1, pp.226–233, Jan./Feb. 2020.

[18] L. Nie et al., "A Reinforcement Learning-Based Network Traffic Prediction Mechanism in Intelligent Internet of Things," *IEEE Transactions on Industrial Informatics,* vol. 17, no. 3, pp.2169–2180, March 2021.

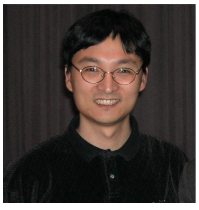
[19] S. P. Arango, F. Heinrich, K. Madhusudhanan, and L. Schmidt-Thieme, "Multimodal meta-learning for time series regression," in *Advanced Analytics and Learning on Temporal Data: 6th ECML PKDD Workshop, AALTD 2021,* pp.123–138, 2021.

[20] AdamPaszke, SamGross, Francisco Massa, AdamLerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. "Pytorch: An imperative style, high-performance deep learning library," in *NeurIPS*, vol. 32, pp. 8026–8037, 2019.

[21] A. Janusz, M. Przyborowski, P. Biczyk and D. lzak, "Network Device Workload Prediction: A Data Mining Challenge at Knowledge Pit," in *Proceedings of the 15th Conference on Computer Science and Information Systems (FedCSIS)*, pp.77–80, 2020.
[22] X. Chen and L. Sun, "Bayesian Temporal Factorization for Multidimensional Time Series Prediction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 9, pp.4659–4673, Sept. 2022.
[23] Z. Cui, R. Ke, and Y. Wang, "Deep Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction," *arXiv* preprint arXiv:1801.02143, 2018.
[24] Z. Cui, K. Henrickson, R. Ke and Y. Wang, "Traffic Graph Convolutional Recurrent Neural Network: A Deep Learning Framework for Network-Scale Traffic Learning and Forecasting," *IEEE Transactions on Intelligent Transportation Systems,* vol. 21, no. 11, pp.4883–4894, Nov. 2020.
[25] S. M. R. Arnold, P. Mahajan, D. Datta, I. Bunner, and K. S. Zarkias, "learn2learn: A Library for Meta-Learning Research," *arXiv* [cs.LG]. http://arxiv.org/abs/2008.12284, 2020.

Ching Chen received her bachelor degree and master degree in Computer Science and Information Engineering from National Taipei University, Taiwan in 2022 and 2024, respectively. She is currently pursuing the Ph.D. degree in electrical engineering and computer science at National Taipei University. Her research interests include machine learning and deep learning.

Hsin-Lung Wu received his Ph.D. degrees in Computer Science and Information Engineering from National Chiao Tung University, Taiwan in 2008. He is currently a Professor in the Department of Computer Science and Information Engineering at National Taipei University, Taiwan. His main research interests include design and analysis of algorithms, computational complexity, theory of machine learning, and deep learning.



Pin-Han Ho (p4ho@uwaterloo.ca) (Fellow, IEEE) is currently a Full Professor in the Department of Electrical and Computer Engineering, University of Waterloo. He is the author/co-author of over 400 refereed technical papers, several book chapters, and the co-author of two books on Internet and optical network survivability. His current research interests cover a wide range of topics information technology and mobile systems, including medical IoT, Space-Aerial-Ground Integrated Networks (SAGINs), and Near-field Telecommunications, etc.

Bor-Shing Lin (M01SM20) received the B.S. degree in electrical engineering from National Cheng Kung University, Taiwan, in 1997, and the M.S. and Ph.D. degrees in electrical engineering from National Taiwan University, Taiwan, in 1999 and 2006, respectively. He is currently a Distinguished Professor with the Department of Computer Science and Information Engineering, and the Director of the Computer Center, National Taipei University, Taiwan. His research interests include smart medicine, embedded systems, wearable systems, biomedical signal processing, biomedical image processing, and portable biomedical electronic system design. He is a fellow of the Institution of Engineering and Technology (IET), U.K., and the British Computer Society (BCS), U.K.