Digital Twin Aided Product Design Framework For IoT Platforms

Chenyu Wang, *Graduate Student Member, IEEE*, Yingshu Li, *Senior Member, IEEE*

***Abstract*—The increasing number of products is the trend of current industry. However, the product development process is significantly limited by budget and testing risk. Recently, *digital twin* has emerged as a promising industrial paradigm that provides an integrated and cohesive view of the product design process. In this paper, we propose a product design framework for Internet of Things (IoT) platforms, namely Digital Twin-aided IoT Platform Design (DTIPD). This framework considers a large number of IoT devices performing different tasks with machine learning (ML) technologies. Each IoT device constructs a particular ML-based model that deals with its task automatically by feeding related data with labels. The challenges of large-scale network management and ground truth shortage at the initial stage of product iteration are addressed. We propose a two-level hierarchical learning process using the real-time model status stored at Digital Twin Servers (DTS), aiming to improve product quality while shortening the development lifecycle. The comprehensive experimental results for both the single-DTS and multiple-DTS scenarios demonstrate the applicability of our framework.**

***Index Terms*—Digital twin, distributed learning, machine learning, internet of things, product design**

I. Introduction

THE INCREASING number of products is the trend and key competitiveness of the industry. It has been reported by Cisco that there will be 29.3 billion connected devices by 2023, up from 18.4 billion in 2018 [1], which has fundamentally reshaped the structure and morphology of the current industrial environment. Some concepts, such as Industry 4.0 [2] and smart manufacturing [3], have been raised to emphasize the importance of product connectivity and integration. Vertical and horizontal process integration is expected to achieve higher performance in the new industrial era.

Driven by the Internet of Things (IoT) and the new wireless networking technologies (e.g., 5G), the deployments of products are more ubiquitous. The application scenarios of industrial products can be either at the macroscopic level such as smart city with big-data analysis and processing [4], [5], or microscopic and personalized recommendation taking place in everyone's smart devices [6]. Leveraging the strength of data-driven and auto-improving algorithms, colloquially named as Machine Learning (ML), the products from IoT platforms have become smarter and more powerful than ever before. The connected products can learn from each other, thus forming a group of more intelligent agents with sufficient exploitation.

Product designers, who give birth to stable products (i.e., accessory equipment, operating supplies or services), are generally responsible for the entire lifecycle of products, including the design, polish and test phases. It is inevitable that designers should sustainably improve and interact with the production system to satisfy the stringent requirements in terms of robustness, efficiency and budget. This is especially important at the initial stage of product design due to the scarce or partially available resources. For this reason, we should comprehensively examine every in-depth change of products led by computing power, intelligent control, and connectivity [7] so as to increase system utility.

One more fact about product design is that tuning products in real-world systems is often risky and costly for many unanticipated damages and errors. Digital Twin (DT), as a promising industrial paradigm, assists product design in a more efficient and responsive way [8]. Often referred to as the comprehensive virtual representation of a group of physical components, DT is invented to collect the data of products throughout all the lifecycle phases. With the virtual replicas of a set of entities powered by the new-generation communication technology, DT is able to exhibit the potential evolution of an industrial system. For example, an ultrahigh fidelity DT model of individual aircraft in [9] is utilized to predict the life of aircraft structure and assure its structural integrity. Moreover, DT enables us to find out more flexible and economic solutions to improve the physical performance of systems in all the lifecycle phases. The General Electric (GE) company developed a DT interface [10] to manage a wind farm, which configures the control features to optimize the performance of a plurality of wind turbines. In addition, DT-aided architectures facilitate troubleshooting remote equipment, thus mitigating system damage or degradation [11]. Inspired by these innovations, we raise the question - whether the DT technology could facilitate the design of ML-based IoT platform products.

In this paper, we propose a DT-aided product design framework for IoT platforms, where the IoT devices are distributed in a network to cope with real-world problems (i.e., classification tasks) using ML-based models. In most scenarios with heterogeneous deployment, the goals of IoT devices can vary significantly by place and time, which raises the requirement of online learning. In summary, the challenges of designing such a DT-aided IoT platform lie in the following aspects:

1) In industrial network environments, it is not realistic to

This work was supported by the National Science Foundation of U.S. under Grant 1829674, and Grant 1741277.

C. Wang and Y. Li are with the Department of Computer Science, Georgia State University, 25 Park Place, Atlanta, GA, 30303, USA (e-mail: cwang50@student.gsu.edu; yili@gsu.edu)

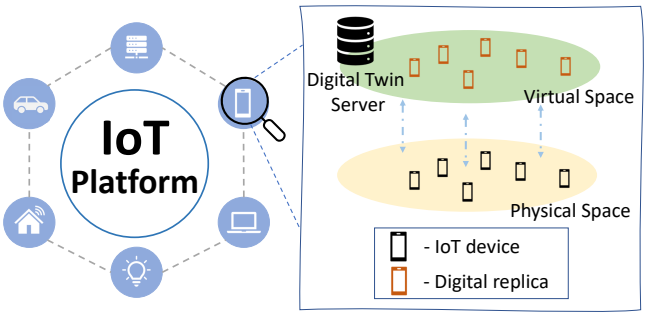

Fig. 1: An IoT platform with the digital twin technology.

synchronize the digital replicas of a considerable number of heterogeneous IoT devices with a single server. Tasks might differ on devices, but we could still leverage some similarities of tasks by transferring the experience of one particular device to others to guide their learning processes.

2) At the initial stage of product design, designers do not have enough resources to label the dataset of a large volume. The manageable size of the dataset with ground truth heavily depends on the budget and human resources. Notably, the unlabeled data can be relatively easier to obtain by IoT devices than the labeled data. For this reason, it is critical to utilize all accessible resources to reduce the development cycle and train reasonable problem-solving models.

To enhance the ML-based product design on a DT-aided IoT platform and address the aforementioned challenges, we propose a DT-aided IoT Platform Design (DTIPD) framework. DTIPD characterizes the process of learning and models the deployment for all the IoT devices in a group of industrial networks. An overview of DTIPD is illustrated in Fig.1, where the IoT platform consists of heterogeneous IoT devices (e.g., smartphones, cars, or sensors) with different tasks sampled from some unknown distribution. For ease of management, the IoT devices dealing with the same task in the physical space are grouped under the same DT Server (DTS) in the virtual space, and they synchronize with the server to update their digital replicas. Each IoT device can only communicate with one DTS, and each DTS serving as a cluster head communicates with the adjacent servers. The digital replicas stored at any DTS are a set of model parameters learned by adjacent DTSs and all the IoT devices regarding the task. For each task, the robust model is iteratively gathered at its corresponding DTS by utilizing the fine-grained models from IoT devices and the features of models from adjacent DTSs. Our DTIPD framework can well address the challenges by decoupling the network through transforming a distributed network to a two-level-hierarchy structure (challenge 1), and for a specific task, utilizing both labeled and unlabeled data from devices, and even the features of models from other tasks at the initial stage to improve data exploitation (challenge 2).

The main contributions of our paper are summarized as follows.

- We formalize the industrial product iterative design for ML-based IoT platforms, in which IoT devices are deployed with ML models to deal with real-world tasks.
- In our framework, DTSs collect real-time status of other components. With DT, we could exert extensive simulations and utilize the results to facilitate product design. To the best of our knowledge, this is the first work applying the DT technology on IoT platforms with a concrete product design pipeline, which is meaningful for future designs with the DT technology.
- In the deployment, two-level hierarchical learning is performed, which includes *task-level learning* and *network-level learning*. DTSs, as the junctions of two-level learning, guide each device using other devices' experience and help to combat the challenge with a few labeled data at the initial stage of product iteration. Moreover, each DTS collaboratively communicates with other DTSs to incorporate the experience of different tasks.
- We discuss the deployment of DTIPD in practice. The experimental results in both single-DTS and multiple-DTS scenarios demonstrate the benefits of our design.

The remainder of this paper is organized as follows. Section II reviews some related works. In Section III, we present the framework with a single DTS. In Section IV, the learning process with multiple DTSs is illustrated. The case studies and experimental results are presented in Section V. Section VI concludes the paper.

II. Related Works

A. Digital Twin

Digital twin (DT), different from traditional technologies such as *Cyber-Physical System* (CPS) [12], [13], [14] and *simulation* [15], [16], was designed under current inevitable trends of Internet of everything, high-capacity computing and network environment. Generally, CPS emphasizes the architecture, where physical and cyber components rely on the information technology for monitoring and control. Simulation is broadly used as an offline tool to analyze complex systems with pre-defined modelings and settings.

Compared with CPS and simulation, the essence of DT is its outcome of constant iterations in the product creating and engineering activities. The work in [17] introduces the methods of finding the best scheduling policy of sensors for IoT applications with the setting of DT, where DT allows service providers to access the real-time cached copies of the sensor readings. An architecture of DT edge network was proposed in [18], where the DTs of edge servers and mobile edge computing environment can coordinate with each other to provide training data for offloading decisions. In summary, DT differs from the traditional computer-aided design and engineering in many ways [19], such as its unique advantages of constant refinement facilitation, traceability promotion and remote troubleshooting. In our work, the model status of IoT devices is constantly monitored by DTSs, which assures the traceability. Moreover, DTSs do not apply the gathered models to local devices directly but provide them with improvement suggestions, which remotely facilitates troubleshooting devices.

B. Distributed Online Learning

Distributed online learning, as a decentralized learning method, updates all the models progressively as the data samples arrive. An online distributed learning method with fixed per-round computing time was proposed in [20]. The work in [21] considers the heterogeneity of distributed networks and enabled tasks offloading during training. The work in [22] devised an iterative distributed learning method, which performs local computation and global communication so as to average the information and synchronize the prediction in an online manner. The aforementioned distributed solutions might not be appropriate for the management of DT-aided IoT platforms, since the tasks of devices differ and the communication mainly takes place between devices and their corresponding DTSs. Moreover, none of these works can address the issue of inadequate labels in product iterations.

C. Unsupervised Learning and Semi-Supervised Learning

In product design, it is essential to seek a way to alleviate the pressure of lacking labeled data at the initial stage of development. Unsupervised learning and semi-supervised learning work promisingly to address the label shortage in the learning process.

In [23], a variety of tasks constructed based on clustering embeddings are trained under the unlabeled data, and the result shows that the final model is applicable to a wide range of tasks. In [24], a principled unsupervised learning model, using Variational Autoencoder (VAE) and set-level variation inference, was proposed to generalize across different tasks without pre-defined task distributions.

Apart from the extreme cases without any labels, some studies considered exploiting some unlabeled data to reduce the overfitting. The work in [25] maintained average model weight target during training process and penalized the predictions that deviate from the target. The work in [26] proposed a semi-supervised learning method which arguments the unlabeled data and used mixed labeled and unlabeled data with label guesses to train the model.

However, none of these methods are considered along with the DT technology. In our work, with DTSs, the digital replicas of both devices and servers are aggressively leveraged to enhance semi-supervised learning.

III. The Single-DTS Scenario

In this section, we start the discussion with an IoT platform where a set of IoT devices, denoted by $\mathcal{N} = \{1, 2, \cdots, N\}$, are deployed in a physical space and directly solve the same image classification task. These devices are grouped under a DTS for ease of management.

To make decisions without being explicitly programmed, IoT devices are asked to train their ML-based models locally according to data samples (e.g., training images) with the ground truth (i.e., true labels). Given that the data are sampled from an unknown distribution $\mathcal{P}$ and the complete label set is $\mathcal{C}$, the ground truth of any data sample can be encoded by a one-hot vector of dimensionality $|\mathcal{C}|$. Each data point has $|\mathcal{C}|$ optional labels and only one of the labels is true. In the product design process, each device first samples a data pool without ground truth from $\mathcal{P}$, and then picks a portion of them to seek the true labels. Assuming that the IoT platform has pre-defined the architecture of ML-based models, the status of any device can be represented by the parameters of its ML-model.

In fact, it is not sufficient to train an applicable model for classification only depending on the strength of a single device, considering some constraints such as storage or computation [27]. One possible solution could be maintaining continuous connections with other devices in the IoT platform to leverage their expertise. However, the cost of devices increases due to the extra computation, and the communication among devices also increases the cyber-security risks [28] and hinders the traceability.

To overcome these drawbacks, we assume that each IoT device does not need to communicate with other devices. With the setting of a DTS, each IoT device can simply upload its model parameters and data samples collected locally via the dedicated channel connecting with its corresponding DTS, thus creating a digital replica in the DTS.

A. Task-Level Learning with Labeled and Unlabeled Data

We first introduce the product design of task-level learning process performed locally by devices. This step mainly takes place in the physical space, where each device incorporates its own data samples for task fitting and generalization. To better formulate the learning process, we divide the whole time horizon of iteration into T time periods. The model deployed in the physical space, denoted by $\boldsymbol{\theta}$, should be gradually improved with time due to the increasing number of data samples and iterations.

At the beginning of any time period $t \in \{1, 2, \cdots, T\}$, each IoT device collects some labeled data samples and uploads them to its DTS. After receiving the labeled data samples from all the IoT devices within the pre-allocated time slot, a DTS constructs a shared labeled dataset $\mathcal{D}_t$. Since the size of the labeled dataset is relatively small, it is realistic to distribute $\mathcal{D}_t$ to all the IoT devices managed by this DTS.

For each time period t and any device $n \in \mathcal{N}$, the learning iterates for $m_{t,n}$ times. In any iteration $i \in [m_{t,n}]$ of time period t, device n samples a minibatch of labeled data points $\mathcal{D}_{t,n}^i$ from the shared labeled dataset $\mathcal{D}_t$, and each point j satisfies $(\boldsymbol{x}_{t,n}^j, y_{t,n}^j) \in \mathbb{R}^d \times \mathcal{C}$, where $\boldsymbol{x}_{t,n}^j$ is the input vector of an image and $y_{t,n}^j$ is its true label. Given the ML-based model architecture, which is pre-defined by the IoT platform, we denote the logits $\boldsymbol{g}(\boldsymbol{x}|\boldsymbol{\theta})$ as the outputs of the last fully connected layer using input vector $\boldsymbol{x}$ and model parameters of $\boldsymbol{\theta}$.

To measure the generalization capability of local model $\boldsymbol{\theta}_{t,n}^i$ within iteration i, device n calculates the cross entropy loss of $\mathcal{D}_{t,n}^i$ using

$$l(\boldsymbol{\theta}_{t,n}^i|\mathcal{D}_{t,n}^i) = \frac{-1}{|\mathcal{D}_{t,n}^i|} \sum_{j \in \mathcal{D}_{t,n}^i} \sum_{c \in \mathcal{C}} p_{t,n}^{j,c} \log q_{t,n}^{j,c}, \tag{1}$$

where $\boldsymbol{p}_{t,n}^{j} = \{p_{t,n}^{j,c}, \forall c \in \mathcal{C}\} = \mathbf{1}_{\{c=y_{t,n}^{j}\}}$ is the one-hot ground-truth label distribution, and $\boldsymbol{q}_{t,n}^{j} = \{q_{t,n}^{j,c}, \forall c \in \mathcal{C}\}$ is calculated with the softmax of logits $\boldsymbol{g}(\boldsymbol{x}_{t,n}^{j}|\boldsymbol{\theta}_{t,n}^{i})$, i.e.,

$$q_{t,n}^{j,c} = \frac{\exp(g^{c}(\boldsymbol{x}_{t,n}^{j}|\boldsymbol{\theta}_{t,n}^{i}))}{\sum_{k\in\mathcal{C}}\exp(g^{k}(\boldsymbol{x}_{t,n}^{j}|\boldsymbol{\theta}_{t,n}^{i}))}. \tag{2}$$

The classification loss $L_{t,n}^{CL}$ of a minibatch of labeled data points $\mathcal{D}_{t,n}^{i}$ is defined as

$$L_{t,n}^{CL}(\boldsymbol{\theta}_{t,n}^{i}|\mathcal{D}_{t,n}^{i}) = l(\boldsymbol{\theta}_{t,n}^{i}|\mathcal{D}_{t,n}^{i}). \tag{3}$$

Meanwhile, device n collects a group of unlabeled dataset $\mathcal{U}_{t,n}^{i}$ in the i-th iteration, where each point j is presented by $\boldsymbol{x}_{t,n}^{j} \in \mathbb{R}^{d}$. The overall loss $f(\boldsymbol{\theta}_{t,n}^{i})$ should be the summation of classification losses for $\mathcal{D}_{t,n}^{i}$ and $\mathcal{U}_{t,n}^{i}$, that is

$$f(\boldsymbol{\theta}_{t,n}^{i}) = L_{t,n}^{CL}(\boldsymbol{\theta}_{t,n}^{i}|\mathcal{D}_{t,n}^{i}) + L_{t,n}^{CL}(\boldsymbol{\theta}_{t,n}^{i}|\mathcal{U}_{t,n}^{i}). \tag{4}$$

However, the ground-truth labels of unlabeled datasets are not commonly accessible if without yielding budget in the product design process, and in our case, the challenge is how to utilize the unlabeled dataset $\mathcal{U}_{t,n}^{i}$ without labeling them.

In practice, since the ground-truth label distributions $\{\boldsymbol{p}\}$ of $\mathcal{U}_{t,n}^{i}$ are missing, the loss $L_{t,n}^{CL}(\boldsymbol{\theta}_{t,n}^{i}|\mathcal{U}_{t,n}^{i})$ in (4) is not available if calculated similarly to (1). Thus, the challenge of task-level training can be concluded as how to train the model if only applying inputs with limited labels, and how to quantify the generalization performance of a model on the unlabeled dataset. To characterize the effect of $\boldsymbol{\theta}_{t,n}^{i}$ on an unlabeled dataset $\mathcal{U}_{t,n}^{i}$, we define a consistency loss $L_{t,n}^{CO}$ instead of $L_{t,n}^{CL}$ for $\mathcal{U}_{t,n}^{i}$ with the aid of digital replicas and a DTS, which is explained in Section III-B.

B. Communication between IoT Devices and DTS

We next introduce how to promote the level of model generalization with a DTS. In general, in the virtual space, a DTS creates the digital replicas of models for all the devices under this server, and each digital replica synchronizes with its corresponding device in the physical space after each step of model iteration.

In details, after receiving the labeled data samples from IoT devices in the data-collecting time slot of time period t, a DTS first initializes a teacher model as $\hat{\boldsymbol{\phi}}_{t}^{1}$, which will guide the learning process of devices under this server. A DTS also creates the real-time model parameter vector of all the IoT devices $\hat{\Theta}_{t} = \{\hat{\boldsymbol{\theta}}_{t,n}\}_{\mathcal{N}}$ as the digital replicas of the physical devices. Both the parameters of teacher model and digital replicas can be written to a real-time table $\Psi_{t} = \{\hat{\boldsymbol{\phi}}_{t}, \hat{\Theta}_{t}\}$, and a DTS initially sets $\hat{\boldsymbol{\phi}}_{t} = \hat{\boldsymbol{\phi}}_{t}^{1}$ and $\hat{\boldsymbol{\theta}}_{t,n} = \boldsymbol{\theta}_{t,n}^{1}, \forall n \in \mathcal{N}$.

As discussed previously, within iteration $i \in [m_{t,n}]$, apart from collecting an unlabeled dataset $\mathcal{U}_{t,n}^{i}$, device n also samples a portion of labeled dataset $\mathcal{D}_{t,n}^{i}$ from the shared labeled dataset $\mathcal{D}_{t}$, and uses $\mathcal{D}_{t,n}^{i} \bigcup \mathcal{U}_{t,n}^{i}$ to iterate its local model $\boldsymbol{\theta}_{t,n}^{i}$. For the labeled dataset, device n can simply derive the classification loss $L^{CL}(\boldsymbol{\theta}_{t,n}^{i}|\mathcal{D}_{t,n}^{i})$ using the loss function defined in (3).

Instead of calculating the classification loss for unlabeled set, we use the consistency loss $L_{t,n}^{CO}$ to measure the ability of local model $\boldsymbol{\theta}_{t,n}^{i}$. To compute the consistency loss, device n first communicates with a DTS to upload $\mathcal{U}_{t,n}^{i}$. Then, this DTS can exert the real-time table Ψ_{t}, including the teacher model $\hat{\boldsymbol{\phi}}_{t}$ and digital replicas $\{\hat{\boldsymbol{\theta}}_{t,n}\}_{\mathcal{N}}$, on batch $\mathcal{U}_{t,n}^{i}$ with a divergence function $Div(\cdot)$ [25], [29], which can be either Mean Squared Error (MSE),

$$D_{MSE}(\boldsymbol{\mu}, \boldsymbol{\nu}) = \frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}}(\mu_{c} - \nu_{c})^{2},$$

or Kullback-Leibler (KL) divergence,

$$D_{KL}(\boldsymbol{\mu}, \boldsymbol{\nu}) = \sum_{c\in\mathcal{C}}\mu_{c}\log\frac{\mu_{c}}{\nu_{c}}.$$

More concretely, a DTS applies the digital model replicas of all the devices except device n to $\mathcal{U}_{t,n}^{i}$, and calculates the logits $\boldsymbol{g}(\boldsymbol{x}_{t,n}^{j}|\hat{\boldsymbol{\theta}}_{t,n'})$ of every data point $j \in \mathcal{U}_{t,n}^{i}$ using models $\{\hat{\boldsymbol{\theta}}_{t,n'}\}_{\mathcal{N}\setminus n}$. With the logit prediction $\{\boldsymbol{g}\}$ of all other devices, one can easily count the predicted distribution for $\mathcal{U}_{t,n}^{i}$ of device n', $\mathbf{q}_{\mathbf{t},\mathbf{n}'}^{\mathbf{i}} = \{\boldsymbol{q}_{t,n'}^{j}, \forall j \in \mathcal{U}_{t,n}^{i}\}$, where $\boldsymbol{q}_{t,n'}^{j} = \{q_{t,n'}^{j,c}, \forall c \in \mathcal{C}\}$, and $q_{t,n'}^{j,c}$ is calculated similarly to (2) with $\boldsymbol{x}_{t,n}^{j}$ of $\mathcal{U}_{t,n}^{i}$. In the same way, we denote $\mathbf{q}_{\mathbf{t},\mathbf{n}}^{\mathbf{i}}$ the predicted distribution of device n using model parameters of $\hat{\boldsymbol{\theta}}_{t,n}$ or $\boldsymbol{\theta}_{t,n}^{i}$.

Hereby, the evaluation of model $\boldsymbol{\theta}_{t,n}^{i}$ for $\mathcal{U}_{t,n}^{i}$ from digital replicas of other IoT devices is equivalent to $Div(\boldsymbol{\mu}_{\mathbf{t},\mathbf{n}}^{\mathbf{i}}, \boldsymbol{\nu}_{\mathbf{t},\mathbf{n}}^{\mathbf{i}})$, where $\boldsymbol{\mu}_{\mathbf{t},\mathbf{n}}^{\mathbf{i}} = \frac{1}{N-1}\sum_{n'\in\mathcal{N}\setminus n}\mathbf{q}_{\mathbf{t},\mathbf{n}'}^{\mathbf{i}} = \{\boldsymbol{\mu}_{t,n}^{j}, \forall j \in \mathcal{U}_{t,n}^{i}\}$ and $\boldsymbol{\nu}_{\mathbf{t},\mathbf{n}}^{\mathbf{i}} = \mathbf{q}_{\mathbf{t},\mathbf{n}}^{\mathbf{i}} = \{\boldsymbol{\nu}_{t,n}^{j}, \forall j \in \mathcal{U}_{t,n}^{i}\}$, and

$$Div(\boldsymbol{\mu}_{\mathbf{t},\mathbf{n}}^{\mathbf{i}}, \boldsymbol{\nu}_{\mathbf{t},\mathbf{n}}^{\mathbf{i}}) = \frac{1}{|\mathcal{U}_{t,n}^{i}|}\sum_{j\in\mathcal{U}_{t,n}^{i}} D_{*}(\boldsymbol{\mu}_{t,n}^{j}, \boldsymbol{\nu}_{t,n}^{j})$$

where D_{*} stands for either MSE or KL divergence.

Similarly, the evaluation of model $\boldsymbol{\theta}_{t,n}^{i}$ for $\mathcal{U}_{t,n}^{i}$ from the teacher model $\hat{\boldsymbol{\phi}}_{t}$ is equivalent to $Div(\hat{\boldsymbol{\mu}}_{\mathbf{t}}^{\mathbf{i}}, \boldsymbol{\nu}_{\mathbf{t},\mathbf{n}}^{\mathbf{i}})$, where $\hat{\boldsymbol{\mu}}_{\mathbf{t}}^{\mathbf{i}}$ includes the softmax of logits for every $j \in \mathcal{U}_{t,n}^{i}$ using $\boldsymbol{g}(\boldsymbol{x}_{t,n}^{j}|\hat{\boldsymbol{\phi}}_{t})$. Hence, the consistency loss for $\mathcal{U}_{t,n}^{i}$ calculated with Ψ_{t},

$$L_{t,n}^{CO}(\Psi_{t}|\mathcal{U}_{t,n}^{i}) = Div(\boldsymbol{\mu}_{\mathbf{t},\mathbf{n}}^{\mathbf{i}}, \boldsymbol{\nu}_{\mathbf{t},\mathbf{n}}^{\mathbf{i}}) + Div(\hat{\boldsymbol{\mu}}_{\mathbf{t}}^{\mathbf{i}}, \boldsymbol{\nu}_{\mathbf{t},\mathbf{n}}^{\mathbf{i}}),$$

is returned to device n for its further local update.

Once the feedback from the DTS is received, device n will replace the local model parameters of $\boldsymbol{\theta}_{t,n}^{i}$ by

$$\boldsymbol{\theta}_{t,n}^{i+1} = SGD(\boldsymbol{\theta}_{t,n}^{i}, \bar{f}(\boldsymbol{\theta}_{t,n}^{i})) \tag{5}$$

using stochastic gradient descent (SGD) [30], where $\bar{f}(\boldsymbol{\theta}_{t,n}^{i})$ is a surrogate loss of (4) incurred by the inquiry of $L_{t,n}^{CO}(\Psi_{t}|\mathcal{U}_{t,n}^{i})$, denoted by

$$\bar{f}(\boldsymbol{\theta}_{t,n}^{i}) = L_{t,n}^{CL}(\boldsymbol{\theta}_{t,n}^{i}|\mathcal{D}_{t,n}^{i}) + \beta L_{t,n}^{CO}(\Psi_{t}|\mathcal{U}_{t,n}^{i}). \tag{6}$$

β is the weight parameter that qualifies the effect of consistency loss. The updated model parameters of $\boldsymbol{\theta}_{t,n}^{i+1}$ then replace $\boldsymbol{\theta}_{t,n}^{i}$ at local physical device n, and will be uploaded to the digital space (i.e., DTS) to update its digital replica $\hat{\boldsymbol{\theta}}_{t,n}$ in vector Θ_{t}.

Suppose the i-th model iteration of device n incurs the i'-th inquiry of all the devices to the DTS within time period

t. The teacher model $\hat{\boldsymbol{\phi}}_t^{i'+1}$ will incorporate $\boldsymbol{\theta}_{t,n}^{i+1}$ using the exponential moving average (EMA) prediction method [25],

$$\hat{\boldsymbol{\phi}}_t^{i'+1} = \alpha \hat{\boldsymbol{\phi}}_t^{i'} + (1-\alpha)\boldsymbol{\theta}_{t,n}^{i+1}, \tag{7}$$

where α is the smoothing coefficient parameter. Finally, a DTS carries out replacement of teacher model $\hat{\boldsymbol{\phi}}_t = \hat{\boldsymbol{\phi}}_t^{i'+1}$ and digital replica $\hat{\boldsymbol{\theta}}_{t,n} = \boldsymbol{\theta}_{t,n}^{i+1}$ in $\hat{\Theta}_t$. The real-time table $\Psi_t = \{\hat{\boldsymbol{\phi}}_t, \hat{\Theta}_t\}$ is simply updated accordingly for each inquiry of devices.

For a device n, after a total of $m_{t,n}$ times of iteration, the local model $\boldsymbol{\theta}_{t,n}$ evolves according to (5), finally deriving $\boldsymbol{\theta}_{t,n}^{m_{t,n}+1}$, which is the initialization $\boldsymbol{\theta}_{t+1,n}^1$ of the next time period. Similarly, after a total of $m_t = \sum_{n\in\mathcal{N}} m_{t,n}$ times of update from different IoT devices, the teacher model $\hat{\boldsymbol{\phi}}_t$ evolves according to (7), finally deriving $\hat{\boldsymbol{\phi}}_t = \hat{\boldsymbol{\phi}}_t^{m_t+1}$, which is the exact model initialization $\hat{\boldsymbol{\phi}}_{t+1}^1$ of next time period. In this way, each IoT device can implement the task-level learning even with a few labeled data samples by fully utilizing a teacher model and digital model replicas of devices in the product design process.

IV. Peer-review Distributed Learning among multiple DTSs

The goals of IoT devices might vary significantly by place and time in an IoT platform. Considering network heterogeneity and deployment, it is not realistic to manage all the IoT devices by a single DTS. In this section, we generalize the DT-aided IoT Platform Design (DTIPD) by introducing multiple DTSs, each of which communicates with the devices dealing with the same task. Each DTS has its own task, which is different from yet related to that of other DTSs.

Although we can directly apply the task-level learning to each set of DTS and devices, it is still necessary to inspect the in-depth change brought by the deployment of multiple DTSs so that all available resources could be fully utilized. One observation is that, compared with ubiquitous devices, DTSs can be more easily managed in the product design process. We assume that each DTS can exchange data with its adjacent DTSs, so as to increase network connectivity.

To characterize the topology of multiple DTSs, we define the network of an IoT platform as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{1, 2, \cdots, V\}$ denotes the set of DTS nodes and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ denotes the edges among DTSs. The set of IoT devices deployed under a DTS node $v \in \mathcal{V}$ is denoted by $\mathcal{N}_v$. The task of a DTS v and its dominated devices $\mathcal{N}_v$ follows the distribution $\mathcal{P}_v \subset \mathcal{P}, \forall v \in \mathcal{V}$. Fig.2 illustrates an example IoT platform with DTIPD, where DTSs can connect to their adjacent DTSs.

Considering DTS connectivity and task similarity, we propose to incorporate the models dealing with different but related tasks from other DTSs to further address the label shortage issue. In DTIPD, besides devices' task-level learning, the network-level learning is further performed by each DTS to improve the generalization ability of its teacher model. We employ a distributed online learning manner so that each DTS can learn from its neighboring DTSs, especially at the early stage of the design with limited ground-truth knowledge.

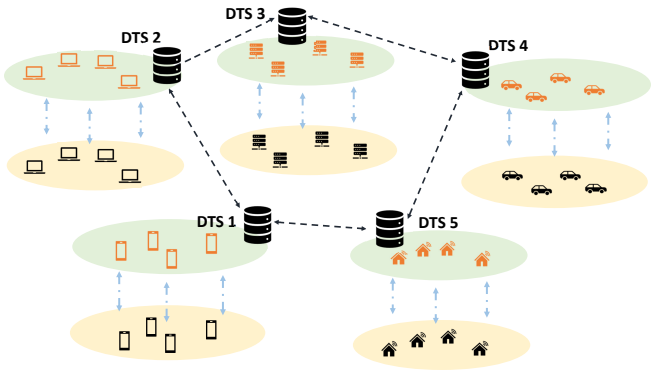

Fig. 2: DTIPD with multiple DTSs.

A. Peer-Review Process at a DTS

As discussed in Section III, at the end of a time period t, each DTS should have completed parameter initialization of teacher model $\hat{\boldsymbol{\phi}}_{t+1,v}^1 = \hat{\boldsymbol{\phi}}_{t,v}^{m_{t,v}+1}$ corresponding to its task[1], which is trained by the task-level learning with its own shared labeled dataset $\mathcal{D}_{t,v}$ and unlabeled dataset

$$\mathcal{U}_{t,v} = \bigcup_{n\in\mathcal{N}_v} \mathcal{U}_{t,n}^1 \cup \cdots \mathcal{U}_{t,n}^{m_{t,n}}.$$

At the network level, each DTS shares its within-task training efforts by broadcasting the teacher model to adjacent DTSs at the beginning of each time period.

The broadcasting process among DTSs can be characterized as following. During the time slot for devices to collect labeled data samples, DTS $v \in \mathcal{V}$ sends its model parameters of $\hat{\boldsymbol{\phi}}_{t,v}^1$ to every adjacent DTS v' in $\mathcal{A}_v = \{v', \forall(v,v') \in \mathcal{E}, v \neq v'\}$, and receives the digital replicas of adjacent DTSs, denoted by $\hat{\Phi}_{t,v} = \{\hat{\boldsymbol{\phi}}_{t,v'} = \hat{\boldsymbol{\phi}}_{t,v'}^1\}_{\mathcal{A}_v}$. To reduce the communication among DTSs, we assume that communication only takes place at the beginning of each time period. By introducing $\hat{\Phi}_{t,v}$, the real-time table of DTS v extends to $\Psi_{t,v} = \{\hat{\boldsymbol{\phi}}_{t,v}, \hat{\Theta}_{t,v}, \hat{\Phi}_{t,v}\}$.

In this way, in addition to the task-level learning, DTS v can perform a peer-review process over its own unlabeled dataset with the evaluation of adjacent nodes' models, which is similar to the online knowledge distillation process in [31]. The results of peer review are used to improve the meta model $\boldsymbol{\phi}_{t,v}$ kept by DTS v. Notably, the parameters of meta model $\boldsymbol{\phi}_{t,v}$ differ from the ones of teacher model $\hat{\boldsymbol{\phi}}_{t,v}$. Besides the parameters of the pre-defined model architecture for classification, the meta model $\boldsymbol{\phi}_{t,v}$ includes two extra projection matrices to identify the importance of other peer models.

In Section III, for any unlabeled data batch $\mathcal{U}_{t,n}^i$ from device $n \in \mathcal{N}_v, i \in [m_{t,n}]$, we only utilize the teacher model of DTS v and digital replicas of other devices in $\mathcal{N}_v$. A DTS evaluates the local model $\boldsymbol{\theta}_{t,n}^i$ by computing the consistency loss $L_{t,v}^{CO}(\Psi_{t,v}|\mathcal{U}_{t,n}^i)$, and returns the result to device n. In generalized DTIPD with multiple DTSs, before using the method introduced in Section III, the attitude of all adjacent peers of DTS v might be collected to update the teacher model $\hat{\boldsymbol{\phi}}_{t,v}$ when v is idle.

[1] $\mathsf{X}_{t,n}$ represents the attributes related to IoT device n and $\mathsf{X}_{t,v}$ represents the attributes related to DTS v.

To incorporate the attitude of peer DTSs into DTS v, we use the meta model $\boldsymbol{\phi}_{t,v}$ to bridge the divisions between $\{\hat{\boldsymbol{\phi}}_{t,v'}\}_{\mathcal{A}_v}$ and $\hat{\boldsymbol{\phi}}_{t,v}$. In details, when DTS v indeterminately chooses a data batch $\mathcal{D}_{t,n}^i \bigcup \mathcal{U}_{t,n}^i$ to train its meta model $\boldsymbol{\phi}_{t,v}$, the evaluation of peer nodes is taken into consideration. After that, the teacher model $\hat{\boldsymbol{\phi}}_{t,v}$ incorporates the meta model $\boldsymbol{\phi}_{t,v}$, hence increasing the generalization ability.

For the labeled dataset $\mathcal{D}_{t,n}^i$, a DTS can simply derive the classification loss of the meta model $L_{t,v}^{CL}(\boldsymbol{\phi}_{t,v}|\mathcal{D}_{t,n}^i)$ using the loss function defined in (3). However, for the unlabeled dataset $\mathcal{U}_{t,n}^i$, we use the consistency loss $L_{t,v}^{CO}$ to measure the ability of the meta model $\boldsymbol{\phi}_{t,v}$ instead of calculating the classification loss. The consistency loss is an overall evaluation of all the other peer-adjacent DTSs.

To be more clear, DTS v can apply the digital replicas in $\hat{\Phi}_{t,v}$, and calculate the logits of every data point $j \in \mathcal{U}_{t,n}^i$ using every adjacent node's model in $\{\hat{\boldsymbol{\phi}}_{t,v'}\}_{\mathcal{A}_v}$, i.e., $\boldsymbol{g}(\boldsymbol{x}_{t,n}^j|\hat{\boldsymbol{\phi}}_{t,v'})$. With the logit prediction $\{\boldsymbol{g}\}$ of all the adjacent DTSs of node v, the predicted distribution for $\mathcal{U}_{t,n}^i$ of each adjacent peer v' is $\mathbf{q}_{\mathbf{t},\mathbf{v}'}^{\mathbf{i},\mathbf{v}} = \{\boldsymbol{q}_{t,v'}^j, \forall j \in \mathcal{U}_{t,n}^i\}$, where $\boldsymbol{q}_{t,v'}^j = \{q_{t,v'}^{j,c}, \forall c \in \mathcal{C}\}$ and

$$q_{t,v'}^{j,c} = \frac{\exp(g^c(\boldsymbol{x}_{t,n}^j|\hat{\boldsymbol{\phi}}_{t,v'}))}{\sum_{k\in\mathcal{C}}\exp(g^k(\boldsymbol{x}_{t,n}^j|\hat{\boldsymbol{\phi}}_{t,v'}))}. \tag{8}$$

The importance of each peer-adjacent DTS's evaluation differs on time. Thus, it is essential to consider the capabilities of peer nodes and identify the differential efficacy of evaluations. To that aim, one can extract peers' evaluation by applying weight parameters to predicted distributions and aggregating them to $\boldsymbol{\mu}_{\mathbf{t},\mathbf{v}}^{\mathbf{i}} = \{\boldsymbol{\mu}_{t,v}^j, \forall j \in \mathcal{U}_{t,n}^i\}$, where

$$\boldsymbol{\mu}_{\mathbf{t},\mathbf{v}}^{\mathbf{i}} = \sum_{v'\in\mathcal{A}_v} w_{v,v'}^t \mathbf{q}_{\mathbf{t},\mathbf{v}'}^{\mathbf{i},\mathbf{v}}, \tag{9}$$

and $w_{v,v'}^t$ represents the normalized efficacy of node v' compared with the other adjacent nodes of DTS v. The method of evaluating $w_{v,v'}^t$ uses two linear projection matrices, and we will detail it in Section IV-B.

We also denote the predicted distribution of DTS v as $\boldsymbol{\nu}_{\mathbf{t},\mathbf{v}}^{\mathbf{i}} = \mathbf{q}_{\mathbf{t},\mathbf{v}}^{\mathbf{i},\mathbf{v}} = \{\boldsymbol{\nu}_{t,v}^j, \forall j \in \mathcal{U}_{t,n}^i\}$ which is similar to (8) but using the logits of the meta model $\boldsymbol{\phi}_{t,v}$. The evaluation of the meta model $\boldsymbol{\phi}_{t,v}$ for $\mathcal{U}_{t,n}^i$ from digital replicas of adjacent peers is equivalent to $Div(\boldsymbol{\mu}_{\mathbf{t},\mathbf{v}}^{\mathbf{i}}, \boldsymbol{\nu}_{\mathbf{t},\mathbf{v}}^{\mathbf{i}})$, where

$$Div(\boldsymbol{\mu}_{\mathbf{t},\mathbf{v}}^{\mathbf{i}}, \boldsymbol{\nu}_{\mathbf{t},\mathbf{v}}^{\mathbf{i}}) = \frac{1}{|\mathcal{U}_{t,n}^i|}\sum_{j\in\mathcal{U}_{t,n}^i} D_*(\boldsymbol{\mu}_{t,v}^j, \boldsymbol{\nu}_{t,v}^j)$$

using either MSE or KL divergence. Similarly, the evaluation of the meta model $\boldsymbol{\phi}_{t,v}$ for $\mathcal{U}_{t,n}^i$ from the teacher model $\hat{\boldsymbol{\phi}}_{t,v}$ is equivalent to $Div(\hat{\boldsymbol{\mu}}_{\mathbf{t},\mathbf{v}}^{\mathbf{i}}, \boldsymbol{\nu}_{\mathbf{t},\mathbf{v}}^{\mathbf{i}})$, where $\hat{\boldsymbol{\mu}}_{\mathbf{t},\mathbf{v}}^{\mathbf{i}}$ is the softmax of logits calculated by $\boldsymbol{g}(\boldsymbol{x}_{t,n}^j|\hat{\boldsymbol{\phi}}_{t,v})$.

Hence, the overall consistency loss considering both peer review and teacher model is given by

$$L_{t,v}^{CO}(\Psi_{t,v}|\mathcal{U}_{t,n}^i) = Div(\boldsymbol{\mu}_{\mathbf{t},\mathbf{v}}^{\mathbf{i}}, \boldsymbol{\nu}_{\mathbf{t},\mathbf{v}}^{\mathbf{i}}) + Div(\hat{\boldsymbol{\mu}}_{\mathbf{t},\mathbf{v}}^{\mathbf{i}}, \boldsymbol{\nu}_{\mathbf{t},\mathbf{v}}^{\mathbf{i}}),$$

and DTS v updates the meta model parameters of $\boldsymbol{\phi}_{t,v}$ by

$$\boldsymbol{\phi}_{t,v}^* = SGD(\boldsymbol{\phi}_{t,v}, \bar{f}(\boldsymbol{\phi}_{t,v})) \tag{10}$$

using stochastic gradient descent. In (10), $\bar{f}(\boldsymbol{\phi}_{t,v})$ is a surrogate loss incurred by the inquiry of $L_{t,v}^{CO}(\Psi_{t,v}|\mathcal{U}_{t,n}^i)$, denoted by

$$\bar{f}(\boldsymbol{\phi}_{t,v}) = L_{t,v}^{CL}(\boldsymbol{\phi}_{t,v}|\mathcal{D}_{t,n}^i) + \beta L_{t,v}^{CO}(\Psi_{t,v}|\mathcal{U}_{t,n}^i), \tag{11}$$

where β is the weight parameter that qualifies the effect of consistency loss. The model parameters of $\boldsymbol{\phi}_{t,v}^*$ will substitute the ones in $\boldsymbol{\phi}_{t,v}$ for the next-time update of network-level learning.

If the i-th inquiry of device n is selected for the network-level learning, and this i-th inquiry is the i'-th inquiry of all the devices to DTS v, the teacher model $\hat{\boldsymbol{\phi}}_{t,v}^{i'+1}$ first incorporates $\boldsymbol{\phi}_{t,v}^*$ (excluding the projection matrices) and then applies the task-level learning to incorporate $\boldsymbol{\theta}_{t,n}^{i+1}$ using the EMA method, which is

$$\hat{\boldsymbol{\phi}}_{t,v}^{i'+1} = \alpha[\alpha\hat{\boldsymbol{\phi}}_{t,v}^{i'} + (1-\alpha)\boldsymbol{\phi}_{t,v}^*] + (1-\alpha)\boldsymbol{\theta}_{t,n}^{i+1},$$

where α is the smoothing coefficient parameter. DTS v will exert parameter replacements of $\hat{\boldsymbol{\phi}}_{t,v} = \hat{\boldsymbol{\phi}}_{t,v}^{i'+1}$ for the following evaluation.

Finally, at the end of time period t, the meta model parameters of $\boldsymbol{\phi}_{t+1,v}$ are set to $\boldsymbol{\phi}_{t,v}$ with $m'_{t,v}$ times of update, and the teacher model parameters of $\hat{\boldsymbol{\phi}}_{t+1,v}^1$ are set to $\hat{\boldsymbol{\phi}}_{t,v}^{m_{t,v}+1}$ with $m_{t,v}$ times of update. $m'_{t,v}$ and $m_{t,v}$ are the total times of network-level learning and task-level learning within time period t respectively. Note that $m'_{t,v}$ can be a number that differs from $m_{t,v}$ since the network-level learning of a DTS uses undetermined batches according to the workload and computing capacities.

B. Individual Weight Graph Calculation

Due to the divergence of tasks processed by different DTSs, each DTS should take the efficacy of other tasks' models into consideration if seeking to utilize them. However, the efficacy is ambitious to determine by manual methods [32]. We adopt a method similar to the work [31] to explore the compatibility of adjacent nodes' models, but maintaining the projection matrices of each DTS independently.

When DTS v evaluates the unlabeled data batch $\mathcal{U}_{t,n}^i$ using the teacher models of DTSs in $\mathcal{A}_v \bigcup v$, high-level features $\{\boldsymbol{h}\}$ are provided in addition to logits $\{\boldsymbol{g}\}$. These features are described as high-level since they contain more information compared with the logits.

To calculate the individual weight graph, DTS v first applies the teacher models of itself and its adjacent nodes to all input vectors of $\mathcal{U}_{t,n}^i$, thus deriving the high-level features $\mathbf{h}_{\mathbf{t},\mathbf{u}}^{\mathbf{i},\mathbf{v}} = \{\boldsymbol{h}(\boldsymbol{x}_{t,n}^j|\hat{\boldsymbol{\phi}}_{t,u}), \forall j \in \mathcal{U}_{t,n}^i\}, u \in \mathcal{A}_v \bigcup v$. Next, the high-level features are projected into two subspaces

$$\boldsymbol{L}_v = \boldsymbol{W}_{L,v}^{\mathsf{T}}\mathbf{h}_{\mathbf{t},\mathbf{v}}^{\mathbf{i},\mathbf{v}} \quad and \quad \boldsymbol{E}_{v,v'} = \boldsymbol{W}_{E,v}^{\mathsf{T}}\mathbf{h}_{\mathbf{t},\mathbf{v}'}^{\mathbf{i},\mathbf{v}}, v' \in \mathcal{A}_v \tag{12}$$

separately, where $\boldsymbol{W}_{L,v}$ and $\boldsymbol{W}_{E,v}$ are the learned projection matrices maintained by DTS v. With the projection matrices, DTS v can determine the individual weight map $\mathbf{w}_{\mathbf{v}}^{\mathbf{t}} = \{w_{v,v'}^t, v' \in \mathcal{A}_v\}$, where

$$w_{v,v'}^t = \frac{e^{\boldsymbol{L}_v^{\mathsf{T}}e^{\boldsymbol{E}_{v,v'}}}}{\sum_{u\in\mathcal{A}_v} e^{\boldsymbol{L}_v^{\mathsf{T}}e^{\boldsymbol{E}_{v,u}}}} \tag{13}$$

Algorithm 1: DTIPD

Input: Network topology $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ and IoT devices set $\{\mathcal{N}_v\}_{\mathcal{V}}$, iteration rounds T;
Output: teacher model $\hat{\boldsymbol{\phi}}_{T+1,v}, \forall v \in \mathcal{V}$.

Initialization:
Initialize the parameters $\hat{\boldsymbol{\phi}}^1_{1,v}$ and $\boldsymbol{\phi}_{1,v}, \forall v \in \mathcal{V}$.
for $t = 1, 2, \cdots, T$ **do**
  - Stage 1: Time slot for collecting labeled datasets and digital replicas of teacher models
  for *all DTS* $v \in \mathcal{V}$ **do**
    $*$ Each device $n \in \mathcal{N}_v$ collects data samples, finds the ground truth, uploads the labeled dataset $\mathcal{D}_{t,n}$ to DTS v;
    $*$ DTS v queries $\hat{\boldsymbol{\phi}}_{t,v'}$ of adjacent DTS $v' \in \mathcal{A}_v$ and updates $\Psi_{t,v}$;
    $*$ DTS v distributes $\mathcal{D}_{t,v}$ to all devices in $\mathcal{N}_v$;
  - Stage 2: Model updates
  for *all DTS* $v \in \mathcal{V}$ **do**
    while *within time period* t *and device* $n \in \mathcal{N}_v$ *collects* $\mathcal{U}^i_{t,n}$ **do**
      $*$ Device n uploads $\mathcal{U}^i_{t,n}$ to DTS v, and samples a minibatch $\mathcal{D}^i_{t,n}$ from $\mathcal{D}_{t,v}$;
      - Network-level Learning (Optional)
      $*$ DTS v calculates $L^{CL}_{t,v}$ and $L^{CO}_{t,v}$ using $\mathbf{w}^{\mathbf{t}}_{\mathbf{v}}$ and digital replicas of teacher models $\hat{\Phi}_{t,v}$;
      $*$ DTS v obtains the overall loss $\bar{f}(\boldsymbol{\phi}_{t,v})$, updates the meta model $\boldsymbol{\phi}_{t,v}$;
      $*$ DTS v updates $\Psi_{t,v}$;
      - Task-level Learning
      $*$ DTS v returns $L^{CO}_{t,n}$ to device n using digital replicas of local models $\hat{\Theta}_{t,v}$;
      $*$ Device n calculates $L^{CL}_{t,n}$, and obtains the overall loss $\bar{f}(\boldsymbol{\theta}^i_{t,n})$, updates the local model $\boldsymbol{\theta}^i_{t,n}$ and the digital replica $\hat{\boldsymbol{\theta}}_{t,n}$;
      $*$ DTS v updates $\Psi_{t,v}$;
    $*$ DTS v initializes $\hat{\boldsymbol{\phi}}^1_{t+1,v} = \hat{\boldsymbol{\phi}}^{m_{t,v}+1}_{t,v}$ and $\boldsymbol{\phi}_{t+1,v} = \boldsymbol{\phi}_{t,v}$.

is calculated using Embedded Gaussian [33] with normalization.

In this way, we can obtain the individual weight graph for all DTSs, and simply calculate the peer-review evaluation of DTS v (9) by substituting $w^t_{v,v'}$ with (13) to weigh the evaluation of other DTSs. Since the projection matrices are included in the meta model $\boldsymbol{\phi}_{t,v}$, the individual weight map of DTS v evolves with the data batches chosen for the network-level learning. The algorithm of two-level hierarchical learning of DTIPD is summarized in Algorithm 1.

V. Experiments and Evaluation

In this section, we present the evaluation of DTIPD under both the single-DTS and multiple-DTS scenarios. The results of the simulation study show that the deployment of DTIPD greatly benefits distributed IoT platforms.

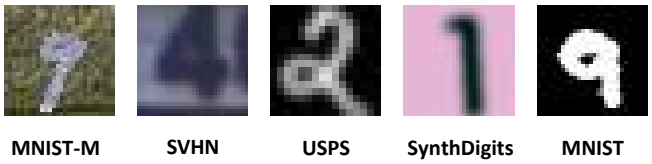

Fig. 3: Data visualization.

Layer	Hyperparameters
Input	28×28 3-channel image
Convolutional	128 filters, 3×3, same padding
Convolutional	128 filters, 3×3, same padding
Convolutional	128 filters, 3×3, same padding
Pooling	Maxpool 2×2
Dropout	$p = 0.5$
Convolutional	256 filters, 3×3, same padding
Convolutional	256 filters, 3×3, same padding
Convolutional	256 filters, 3×3, same padding
Pooling	Maxpool 2×2
Dropout	$p = 0.5$
Convolutional	512 filters, 3×3, valid padding
Convolutional	256 filters, 1×1, same padding
Convolutional	128 filters, 1×1, same padding
Pooling	Average pool ($6 \times 6 \rightarrow 1 \times 1$ pixels)
Softmax	Fully connected $128 \rightarrow 10$

TABLE I: The convolution network architecture used in the experiments.

A. Tasks and Models in Simulation Environments

In practice, there are many kinds of tasks, such as image classification, speech recognition, robotic control, etc. In this work, we consider an IoT platform focusing on image classification tasks, specifically digit classification. As shown in Fig.3, we used five datasets to evaluate the performance of DTIPD. The datasets are MINST-M [34], SVHN [35], USPS [36], SynthDigits [34], and MNIST [37], each of which stands for a kind of task and consists of images of 0-9 digits. Either gray-scale or RGB images are reshaped to 28×28-pixel images with 3 channels, and the pixel values of each image are normalized to a fixed mean and standard deviation. According to the feature distributions of datasets, we can treat the tasks as non-iid [38].

In each task, we used the training set that consists of 7,438 images randomly sampled from the original training dataset and kept the testing set the same as in [38]. As formulated in the previous sections, we used cross-entropy loss between softmax distributions and one-hot labels as the classification loss, and calculated the consistency loss using MSE[2].

In the simulation experiments, we applied a 13-layer Convolution Neural Network (CNN) [25] on all the IoT devices as shown in Table I. In addition to the 13-layer CNN, two linear projection layers are deployed with the meta model of all the DTSs, both of which take the high-level features

[2]We only use the divergence function of MSE, since the performance difference between MSE and KL divergence is outside of the scope of this paper.

after the last pooling operation as input. All digital replicas of DTSs' teacher models or devices' local models only contain the 13-layer CNN.

In each group of experiments, a portion of the training dataset was selected as the labeled dataset while the remaining data points were treated as the unlabeled dataset. In each round of training, we divided the whole unlabeled dataset into many minibatches with size 64. Meanwhile, for each unlabeled minibatch, we randomly sampled a minibatch of 64 data samples from the labeled dataset, and combined this labeled minibatch with the unlabeled minibatch to form a training batch. The total number of combined training batches is the same as the number of the unlabeled minibatches. We ran the SGD optimizer for $T = 180$ epochs with the learning rate $\eta_0 = 0.05$ and maximum consistency weight parameter $\beta = 100$. The EMA weight parameter α is set to 0.97. Similar to [25], the consistency cost parameter β is subject to the ramp-up from 0 to the maximum value, using the sigmoid-shaped function $e^{-5(1-y)^2}$, where $y \in [0, 1]$.

B. Evaluation for the Single-DTS Scenario

In the single-DTS scenario, each DTS does not communicate with other DTSs for teacher model synchronization, thus only consisting of task-level learning. To show the performance of our method in the single-DTS scenario, we employ 5% data points of the training dataset as labeled data (i.e., 371 images with labels), and compute the testing accuracy of the teacher models (EMA models) stored at the DTSs with the following settings:

1) *S5D5-EMA*. There are 5 DTSs, each of which is responsible for 5 IoT devices. In each training iteration, one of the 5 devices is randomly chosen to receive the training batch. The chosen device updates the local model and incurs teacher model update according to the received batch. Both labeled and unlabeled data are used for training.
2) *S5D1-EMA*. The setting is similar to that of S5D5-EMA except that there is only 1 device under each DTS. To compare with S5D5-EMA, each of the training batch can be received by the only device under the DTS with the probability of 20%.
3) *S5D1-NON-EMA*. The setting is similar to that of S5D1-EMA, where there is only 1 device under each DTS. The difference is that each device only utilizes the labeled data in each received training batch, and directly uploads its updated model parameters to DTS. No unlabeled data are utilized in this case.

Fig.4 presents the testing accuracy of the 13-layer CNN models in terms of 5 different datasets with the settings of S5D1-EMA, S5D1-NON-EMA, and S5D5-EMA. As shown in Fig.4, the results in S5D5-EMA outperform the ones in the other two cases in regard of both testing accuracy and convergence speed. The reasons lie in two aspects. The first reason is that unlabeled data samples play important roles in the training process with limited data labels. Table II shows the best testing accuracy of different tasks. According to the values in the rows for S5D1-EMA and S5D1-NON-EMA in Table II, we can find that the use of unlabeled data steadily improves performance in all the different tasks from **0.62%** to **8.57%**, especially for the MINST-M and SVHN datasets, which confirms our perspective - it is critical to leverage unlabeled data at the initial stage of product design. The second reason is related to the number of devices controlled by a DTS. Since we adopt the averaging model at the DTSs, it is of great advantage that each device can simply upload its model parameters to its DTS after each iteration step. The weight average approach scales well to large datasets [25] and more devices, which means the increasing number of active devices can accelerate the development lifecycle of products. This perspective can be validated by the rows for S5D1-EMA and S5D5-EMA in Table II, where by increasing the number of devices from 1 to 5, the performance of different tasks improves from **0.47%** to **6.60%**. The weight averages of model parameters also improve all layer outputs, which is beneficial in the peer-review process with multiple DTSs due to the exploitation of high-level features.

C. Evaluation for the Multiple-DTS Scenario

To show the performance of DTIPD in multiple-DTS scenario, we discuss the deployment of DTSs adopting the following topologies shown in Fig.5: (a) *Isolation* (ISO), (b) *Strong Connectivity* (SC). Each DTS controls 5 devices. In Fig.5(a), each DTS is completely isolated from other DTSs, while in Fig.5(b), each DTS can synchronize the digital replicas of teacher models of all the other DTSs. To explore to what extent a DTS can exploit the models of other DTSs, we consider the following cases: SC-0.02 and ISO-0.02, SC-0.05 and ISO-0.05, SC-0.08 and ISO-0.08, where the number stands for the portion of labeled data used for learning. The peer-review process is not adopted in all the ISO cases. Without loss of generality, in all the SC cases, a DTS only carries out network-level learning for the inquiry raised by device 1, which ensures that the workload of a DTS is compatible with that of devices. The random seeds used for training and sampling are different from the ones in the single-DTS scenario.

Fig.6 presents the testing accuracy of the 13-layer CNN models in terms of 5 different datasets with the settings of SC-0.02, ISO-0.02, SC-0.05, ISO-0.05, SC-0.08 and ISO-0.08. The results in Fig.6 show that the digital replicas of teacher models have inconstant impact on different tasks with various numbers of labeled data samples. We also present the best testing accuracy of different tasks in Table III.

In Fig.6(b), we can observe obvious anomalies in both the SC-0.02 and ISO-0.02 cases of SVHN task. The phenomena are caused by the overfitting since we only apply an extremely limited number of labeled samples in this two cases. Hence, we omit the best accuracy of SVHN task for SC-0.02 and ISO-0.02 in Table III. However, we find that the other tasks can still benefit from the peer-review process in regard of convergence speed or accuracy, according to the results of SC-0.02 and ISO-0.02 shown in Fig.6.

As shown in Fig.6(d) and (e) and from Table III, with 2% of the labeled data at all DTSs, the best testing accuracy with respect to the SynthDigits and MNIST tasks increases

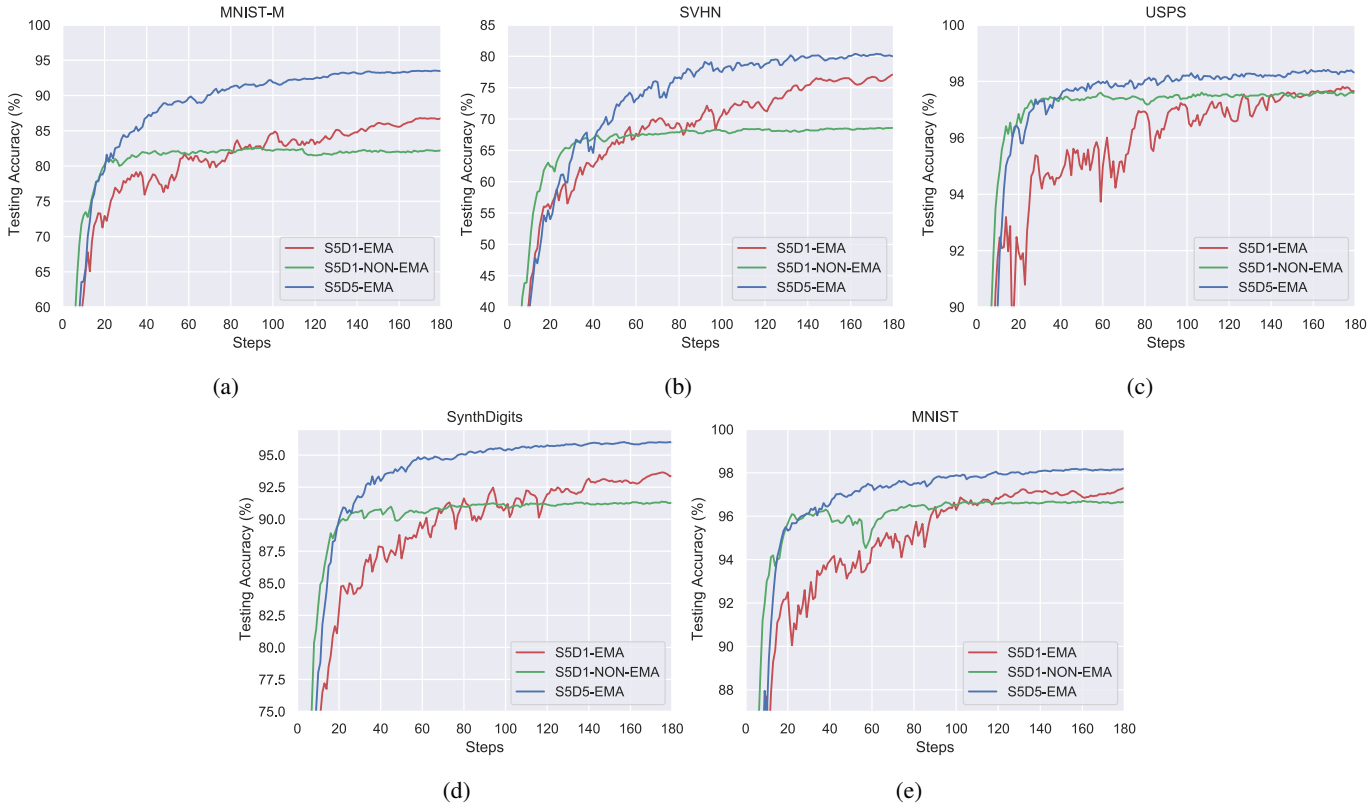

Fig. 4: The testing accuracy of 13-layer CNN models with (a) MNIST-M, (b) SVHN, (c) USPS, (d) SynthDigits, (e) MNIST tasks in the S5D5-EMA, S5D1-EMA and S5D1-NON-EMA cases.

Datasets	MNIST-M	SVHN	USPS	SynthDigits	MNIST
S5D5-EMA	**93.62**	**80.87**	**98.54**	**96.07**	**98.22**
S5D1-EMA	87.02	77.37	98.07	93.71	97.41
S5D1-NON-EMA	82.79	68.80	97.76	91.51	96.79

TABLE II: Best testing accuracy of different tasks (datasets) in the S5D5-EMA, S5D1-EMA and S5D1-NON-EMA cases of the single-DTS scenario.

Datasets	MNIST-M		SVHN		USPS		SynthDigits		MNIST	
Modes	ISO	SC	ISO	SC	ISO	SC	ISO	SC	ISO	SC
148 labeled samples (2%)	81.55	**81.91**	-	-	98.39	**98.44**	93.77	**94.44**	96.90	**97.82**
371 labeled samples (5%)	91.50	**92.93**	83.48	**85.22**	98.33	**98.70**	96.29	96.28	98.38	98.31
595 labeled samples (8%)	94.38	**94.96**	88.96	**89.72**	98.59	**98.80**	97.09	97.07	98.29	**98.62**

TABLE III: Best testing accuracy of different tasks (datasets) in the SC-0.02 and ISO-0.02, SC-0.05 and ISO-0.05, SC-0.08 and ISO-0.08 cases of the multiple-DTS scenario.

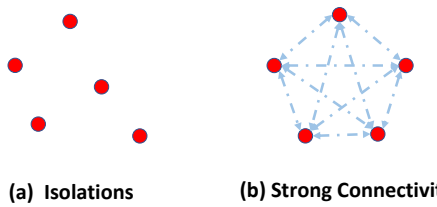

Fig. 5: Two simulation topology cases considered in the experiments.

by **0.67%** and **0.92%** when adopting the DTS peer-reviewing process. As the portion of labeled data samples increases to 5%, we cannot tell the significant difference of testing accuracy for the SC and ISO cases in the SynthDigits and MNIST tasks. Despite this uncertainty, the SC cases start to outperform the ISO cases by a significant margin in the MNIST-M and SVHN tasks, achieving **1.43%** and **1.74%** higher accuracy. When continuously increasing the labeled data samples to 8% of the training dataset, we could still observe the advantages of the peer-review process in the MNIST-M and SVHN tasks since the best testing accuracy with respect to the MNIST-M and SVHN tasks increases by

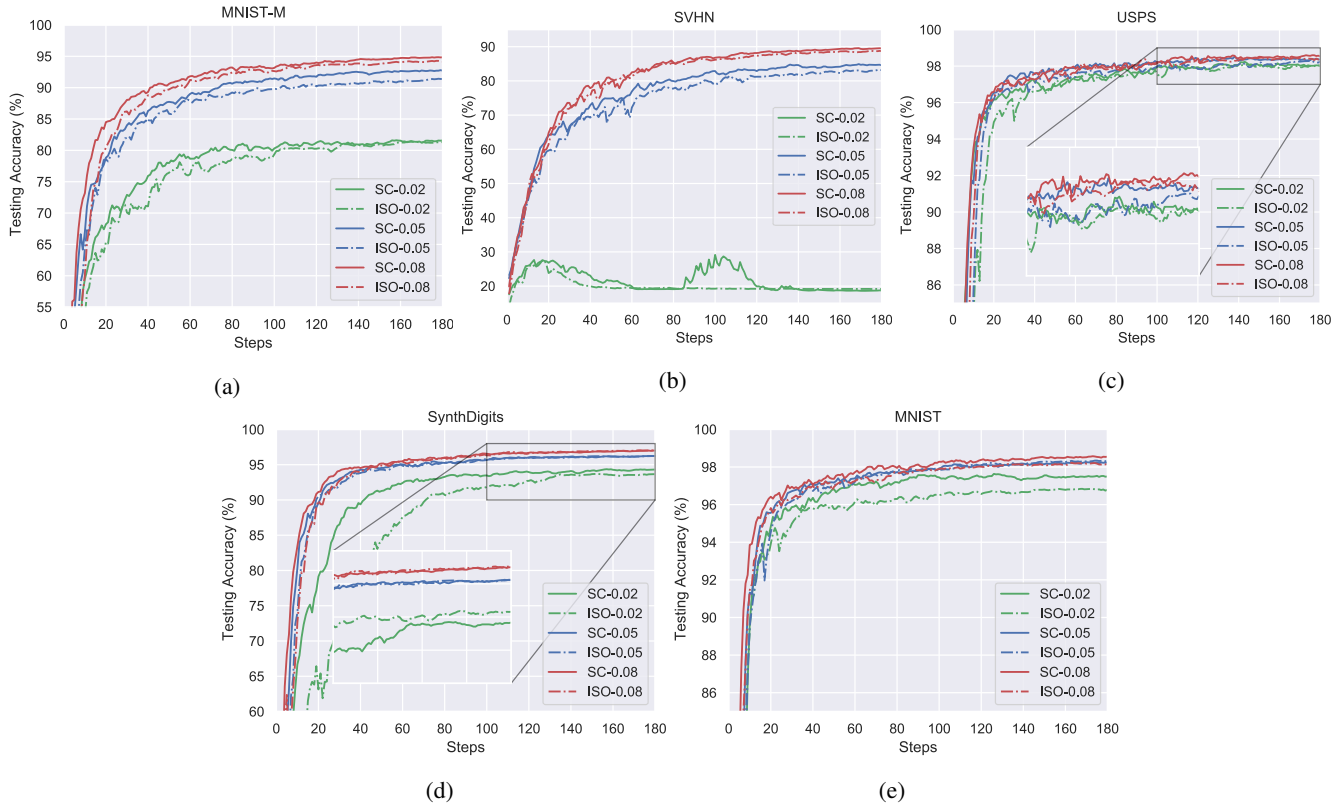

Fig. 6: The testing accuracy of the 13-layer CNN models with (a) MNIST-M, (b) SVHN, (c) USPS, (d) SynthDigits, (e) MNIST tasks in the SC-0.02 and ISO-0.02, SC-0.05 and ISO-0.05, SC-0.08 and ISO-0.08 cases.

0.58% and **0.76%** when using the SC case instead of the ISO case. It is because the benefits of peer-review process might be reduced when supplying sufficient labeled data samples. The peer-review process is significantly sensitive to both the numbers of available labeled data samples and distributions of tasks. This means in practice, we could adjust the network-level learning policy according to the progress of product iterations.

VI. Conclusions

In this paper, we propose a DT-aided product design framework for IoT platforms, namely Digital Twin-aided Internet of things Platform Design (DTIPD). This is to address the issues of large-scale network management and lack of ground-truth labels at the initial product design stage for ML-based IoT platforms. Digital twin, as an emerging paradigm, is promising to improve the future product management and design. The comprehensive experimental results in both the single-DTS and multiple-DTS scenarios demonstrate the applicability of DTIPD.

As future works, we plan to employ the DT technique in a wide range of applications. First, an outstanding feature brought by DT is that it enables customers to visualize the IoT assets in real-time across the entire lifecycle. To this aim, we will study how to promote information synchronization in DT-aided IoT platforms. Second, the work in [39] proposed to build DT to forecast the future state transition of physical entities. However, this direction is not yet well studied. We plan to study the DT-aided fault detection methods to improve the quality of IoT services. Additionally, data or model sharing may cause severe privacy issues in industrial IoT platforms [40]. We will study the data leakage problem when applying the DT technique in real-world production systems.

References

[1] Cisco, "Cisco annual internet report (2018–2023) white paper," https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html, updated March 8, 2020.

[2] L. S. Dalenogare, G. B. Benitez, N. F. Ayala, and A. G. Frank, "The expected contribution of industry 4.0 technologies for industrial performance," *International Journal of Production Economics*, vol. 204, pp. 383–394, 2018.

[3] A. Kusiak, "Smart manufacturing," *International Journal of Production Research*, vol. 56, no. 1-2, pp. 508–517, 2018.

[4] Z. Ullah, F. Al-Turjman, L. Mostarda, and R. Gagliardi, "Applications of artificial intelligence and machine learning in smart cities," *Computer Communications*, vol. 154, pp. 313–323, 2020.

[5] Z. Cai and Z. He, "Trading private range counting over big IoT data," in *Proceedings of the 39th IEEE International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2019, pp. 144–153.

[6] T. Yu, Y. Yang, Y. Li, X. Chen, M. Sun, and P. Li, "Combo-attention network for Baidu video advertising," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 2474–2482.

[7] M. V. P. Pessôa and J. M. J. Becker, "Smart design engineering: a literature review of the impact of the 4th industrial revolution on product design and development," *Research in engineering design*, vol. 31, no. 2, pp. 175–195, 2020.

[8] F. Tao, H. Zhang, A. Liu, and A. Y. Nee, "Digital twin in industry: State-of-the-art," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2405–2415, 2018.

[9] E. J. Tuegel, A. R. Ingraffea, T. G. Eason, and S. M. Spottswood, "Reengineering aircraft structural life prediction using a digital twin," *International Journal of Aerospace Engineering*, vol. 2011, 2011.
[10] A. M. Lund, K. Mochel, J.-W. Lin, R. Onetto, J. Srinivasan, P. Gregg, J. E. Bergman, K. D. Hartling, A. Ahmed, and S. Chotai, "Digital twin interface for operating wind farms," United States, Patent Application Publication, No. 2016/0333854 A1, 2016.
[11] B. Schleich, N. Anwer, L. Mathieu, and S. Wartzack, "Shaping the digital twin for design and production engineering," *CIRP Annals*, vol. 66, no. 1, pp. 141–144, 2017.
[12] S. Sridhar, A. Hahn, and M. Govindarasu, "Cyber–physical system security for the electric power grid," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 210–224, 2011.
[13] A. Villalonga, G. Beruvides, F. Castaño, and R. E. Haber, "Cloud-based industrial cyber–physical system for data-driven reasoning: A review and use case on an industry 4.0 pilot line," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, pp. 5975–5984, 2020.
[14] Z. Cai and X. Zheng, "A private and efficient mechanism for data uploading in smart cyber-physical systems," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 2, pp. 766–775, 2018.
[15] X. Hu and P. Wu, "A data assimilation framework for discrete event simulations," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 29, no. 3, pp. 1–26, 2019.
[16] S. Rai and X. Hu, "Hybrid agent-based and graph-based modeling for building occupancy simulation," in *Proceedings of the 4th ACM International Conference of Computing for Engineering and Sciences*, 2018, pp. 1–12.
[17] L. Corneo, C. Rohner, and P. Gunningberg, "Age of information-aware scheduling for timely and scalable Internet of things applications," in *Proceedings of IEEE Conference on Computer Communications*. IEEE, 2019, pp. 2476–2484.
[18] W. Sun, H. Zhang, R. Wang, and Y. Zhang, "Reducing offloading latency for digital twin edge networks in 6G," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 10, pp. 12 240–12 251, 2020.
[19] A. M. Madni, C. C. Madni, and S. D. Lucero, "Leveraging digital twin technology in model-based systems engineering," *Systems*, vol. 7, no. 1, pp. 7–7, 2019.
[20] N. Eshraghi and B. Liang, "Distributed online optimization over a heterogeneous network with any-batch mirror descent," in *Proceedings of International Conference on Machine Learning*. PMLR, 2020, pp. 2933–2942.
[21] Y. Tu, Y. Ruan, S. Wagle, C. G. Brinton, and C. Joe-Wong, "Network-aware optimization of distributed learning for fog computing," in *Proceedings of IEEE Conference on Computer Communications*. IEEE, 2020, pp. 2509–2518.
[22] K. I. Tsianos and M. G. Rabbat, "Efficient distributed online prediction and stochastic optimization with approximate distributed averaging," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 4, pp. 489–506, 2016.
[23] K. Hsu, S. Levine, and C. Finn, "Unsupervised learning via meta-learning," *arXiv preprint arXiv:1810.02334*, 2018.
[24] L. Dong Bok, M. Dongchan, L. Seanie, and H. Sung Ju, "Meta-GMVAE: Mixture of Gaussian VAE for unsupervised meta-learning," in *Proceedings of International Conference on Learning Representations*, 2021.
[25] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 1195–1204.
[26] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. A. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," in *Proceedings of Advances in Neural Information Processing Systems*, 2019.
[27] J. Zhou, Z. Cao, X. Dong, and A. V. Vasilakos, "Security and privacy for cloud-based IoT: Challenges," *IEEE Communications Magazine*, vol. 55, no. 1, pp. 26–33, 2017.
[28] B. Sudharsan, J. G. Breslin, and M. I. Ali, "Edge2train: a framework to train machine learning models (SVMs) on resource-constrained IoT edge devices," in *Proceedings of the 10th International Conference on the Internet of Things*, 2020, pp. 1–8.
[29] S. Laine and T. Aila, "Temporal ensembling for semi-supervised learning," *arXiv preprint arXiv:1610.02242*, 2016.
[30] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proceedings of the 30th International conference on machine learning*. PMLR, 2013, pp. 1139–1147.
[31] D. Chen, J.-P. Mei, C. Wang, Y. Feng, and C. Chen, "Online knowledge distillation with diverse peers," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 3430–3437.
[32] A. H. Sayed, "Adaptive networks," *Proceedings of the IEEE*, vol. 102, no. 4, pp. 460–497, 2014.
[33] Y. Cao, J. Xu, S. Lin, F. Wei, and H. Hu, "GCNet: Non-local networks meet squeeze-excitation networks and beyond," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019, pp. 0–0.
[34] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *Proceedings of the 32nd International conference on machine learning*. PMLR, 2015, pp. 1180–1189.
[35] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.
[36] J. J. Hull, "A database for handwritten text recognition research," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 16, no. 5, pp. 550–554, 1994.
[37] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[38] X. Li, M. Jiang, X. Zhang, M. Kamp, and Q. Dou, "FedBN: Federated learning on non-iid features via local batch normalization," *arXiv preprint arXiv:2102.07623*, 2021.
[39] C. Zhuang, T. Miao, J. Liu, and H. Xiong, "The connotation of digital twin, and the construction and application method of shop-floor digital twin," *Robotics and Computer-Integrated Manufacturing*, vol. 68, p. 102075, 2021.
[40] X. Zheng and Z. Cai, "Privacy-preserved data sharing towards multiple parties in industrial IoTs," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 5, pp. 968–979, 2020.

Chenyu Wang received his B.S. degree from Xiangtan University, China and M.S. degree from Beijing Normal University, China. He is currently pursuing the Ph.D. degree in the Department of Computer Science at Georgia State University. His research interests include network optimization and machine learning.

Yingshu Li received her Ph.D. and M.S. degrees from University of Minnesota-Twin Cities. She received her B.S. degree from Beijing Institute of Technology, China. Dr. Li is currently a Professor in the Department of Computer Science at Georgia State University. Her research interests include Privacy-aware Computing, Management of Big Sensory Data, Internet of Things, Social Networks, and Wireless Networking. Dr. Li is the recipient of the NSF CAREER Award. Dr. Li has served as an Associate Editor or Guest Editor for some prestigious journals such as ACM Transactions on Sensor Networks, IEEE Transactions on Computers, IEEE Transactions on Network Science and Engineering, and IEEE Internet of Things Journal. She has also served as a Steering Committee Chair, General Chair, Program Chair, and Technical Program Committee member for many international conferences.