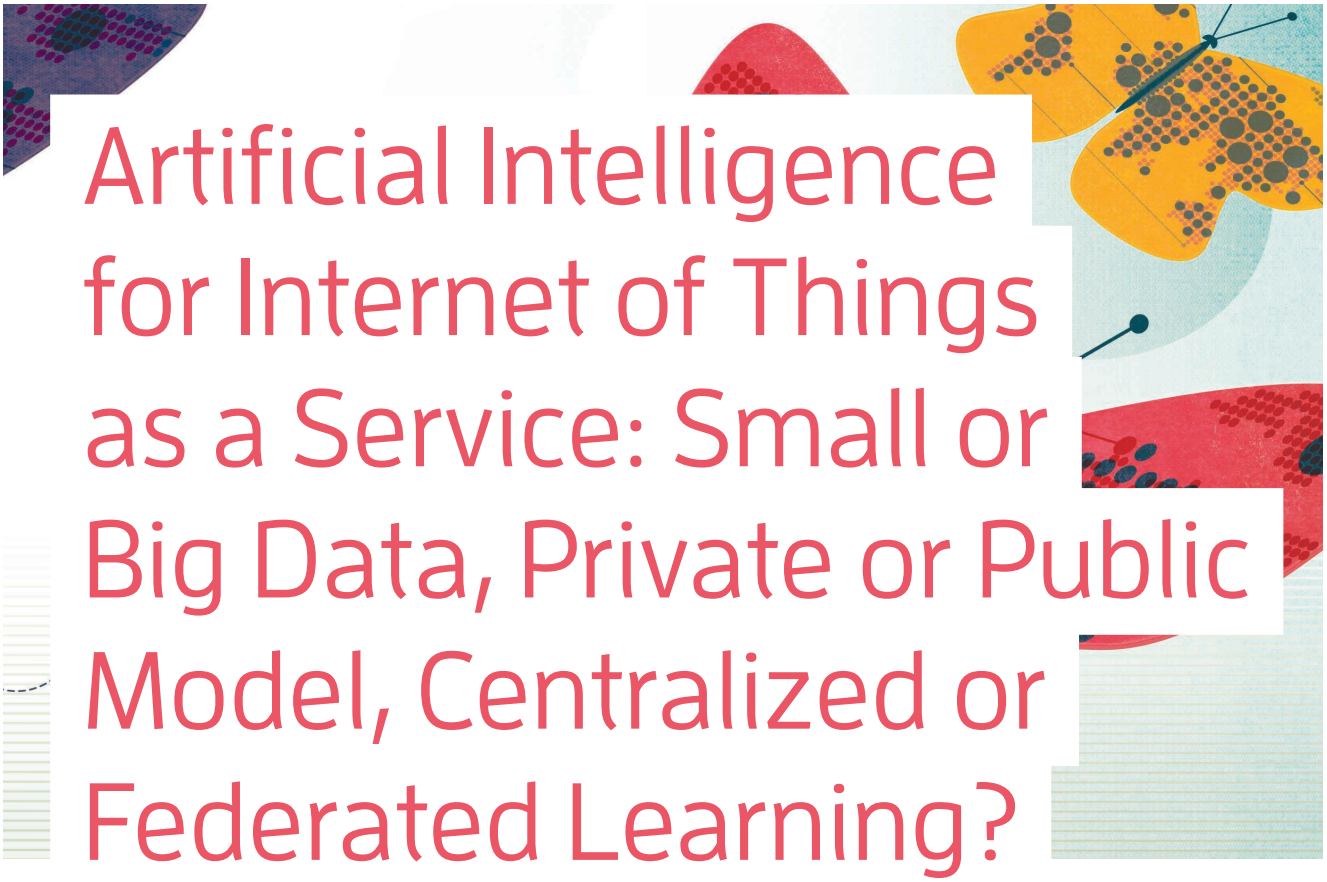

Artificial Intelligence for Internet of Things as a Service: Small or Big Data, Private or Public Model, Centralized or Federated Learning?

Ying-Dar Lin, National Yang Ming Chiao Tung University
Yuan-Cheng Lai, National Taiwan University of Science and Technology
Didik Sudyana and Ren-Hung Hwang, National Yang Ming Chiao Tung University

We propose a generic framework for mapping the training and federation tasks of three artificial intelligence for Internet of Things as a service (AIoTaS) systems to multiple cloud–edge–fog paradigms. A total of 31 possible mappings are identified as possible reference designs for AIoTaS providers.

The Internet of Things (IoT) has been pervasively deployed in recent years and enables the collection of massive amounts of data from a wide variety of sources. However, these data must be properly processed and analyzed to identify the emerging patterns and formulate appropriate subsequent actions. Due to the sheer volume of the data involved, this task is best performed by combining the IoT infrastructure with artificial intelligence (AI) technology. This combination, referred to as *artificial intelligence of things* (*AIoT*), integrates the decision-making power of AI with the connectivity of IoT and has significant advantages for improving efficiency.[1]

INTRODUCTION

IoT owners and developers that employ IoT applications, have many tasks to perform. For example, they must not only develop their IoT applications, but also put them on

Digital Object Identifier 10.1109/MC.2023.3303370
Date of current version: 13 November 2023

the devices and manage them. These tasks impose a significant burden on the capabilities of the owners and developers,[2] particularly when AI technology is involved. Consequently, great interest exists in outsourcing some (or all) of these tasks to service providers, thereby introducing the paradigms of IoT and AIoT as a service (that is, IoTaS and AIoTaS, respectively).

In implementing AIoT, one of the most crucial issues is that of the data required to train the machine learning (ML) model. Broadly speaking, these data can be classified as either *small* or *big* data, depending on the owner and the nature of the model being trained. In particular, small data are owned by a tenant and are used to train a private model, whereas big data are owned by all or some of the tenants and are used to train a public model. Public models have a higher accuracy than private models since they acquire global knowledge from all of the data received from all tenants.[3] However, private models have superior privacy since the data that are trained are not shared.[4] Furthermore, tenants can improve the accuracy of these models by leveraging federated learning to generate global models based on the aggregated private models of different tenants. Notably, such small-public configurations retain the advantage of small-private configurations in that the original data are still kept private.

Another important consideration when deploying AIoT platforms is the service architecture to be employed.[5] Several computing paradigms can be utilized to support AIoT services, including cloud, edge, fog, and a federation of two or more of them. The federation between these computing paradigms can be considered as a bigger computing platform that enables them to collaborate despite being decoupled and operated individually.[6] Note that the concepts of *federation* and *federation learning* (*FL*) are different, where the former forms the integration at the lower platform level through the federation of cloud–edge–fog paradigms while the latter integrates the upper-level ML process. They don't need to coexist. Nevertheless, it is possible to combine both by running FL for AIoTaS over the federated cloud–edge–fog platform. Figure 1 illustrates the general system architecture employing a multitier approach to deliver AIoTaS to tenants.

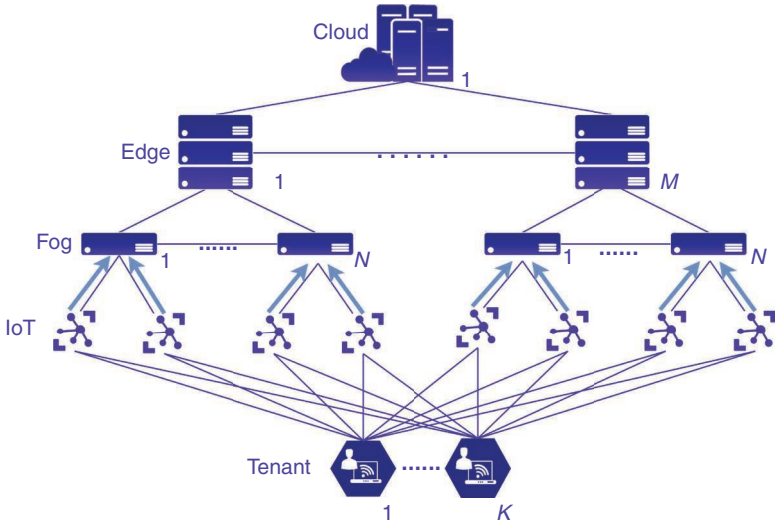

FIGURE 1. Multitier architecture for AIoT system. *M*: number of edge nodes; *N*: number of fogs under each edge.

This article develops a generic framework to guide service providers in determining an appropriate cloud–edge–fog paradigm on which to perform specific AIoTaS services. Three possible service modes are considered, namely *big public*, *small private*, and *small public*. For each service, the corresponding learning and federation tasks are mapped to multiple cloud–edge–fog paradigms. A total of 31 possible mappings are identified. The various mappings are introduced and described, and the open challenges facing the commercialization of AIoTaS are then briefly discussed.

The primary motivation for this research is to explore the various service models within the AIoT framework and understand their rationality and implications. By examining the small-private, big-public, and small-public service models, as well as their integration with various computing paradigms, we aim to provide a comprehensive analysis of the tradeoffs among data privacy, accuracy, and collaboration in the context of AIoTaS. This research addresses a gap in existing literature, where the focus has primarily been on ML models for IoT[7,8,9,10,11] or optimizing IoT architectures through resource management.[12,13,14,15,16,17] We seek to provide a generic AIoTaS framework to service providers across various domains by exploring and discussing the service tradeoffs that have not been adequately covered in previous research or survey papers.[18,19,20,21]

The remainder of this article is organized as follows. The "Related Works" section reviews previous work in the field. The "AIoT Dimension" section explains the main dimensions of AIoT. The "Services on AIoT" section describes three service modes of AIoTaS. The "Multitier Architectures" section discusses the multitier architectures to support AIoTaS, followed by a section presenting the service-architecture mapping results. The "Open Challenges" section describes the open challenges facing the AIoTaS field, and we end with some brief concluding remarks.

RELATED WORKS

This section presents an overview of the existing research conducted in the field of AIoT systems, and Table 1 provides a comprehensive summary of existing research on this field. The majority of these studies focus on the investigation of specific ML models or architectures that can support the complex operations of AIoT, while others provide a survey of the field.

In terms of ML models, researchers have investigated techniques to enhance the accuracy, robustness, the convergence time and the communication efficiency of FL in the context of AIoT.[7,8,9,10,11] These studies have explored approaches to improve not only FL accuracy, but also the convergence time and address the challenges related to model training and data privacy in distributed AIoT environments.

There has also been significant work on the optimization and implementation of architectures suitable for AIoT. These studies primarily consider public models with centralized learning, focusing on issues such as the scalability of AIoT for microservices,[12] offloading optimization to minimize delay,[13] task scheduling optimization to minimize the energy consumption,[14] the development of scalable frameworks to support AIoT services,[15] the intelligence orchestration for AIoT,[16] and the platform for autodeploying AIoT applications.[17]

Additionally, several survey papers have taken a broader view of the AIoT field. These works have reviewed the ML algorithms used in AIoT,[18] explored recent approaches and technologies supporting AIoT,[19] considered the use of edge and cloud collaboration modes to facilitate AIoT,[20] and discuss the progress, challenges, and opportunities of AIoT systems.[21]

Despite these valuable contributions, the existing literature fails to address AIoTaS, an emerging paradigm that combines the as-a-service model with AIoT technologies to offer scalable, flexible, and efficient solutions. Furthermore, a substantial gap remains in terms of a robust examination of the data, model, and job dimensions of AIoT, and the mapping of these dimensions to suitable architectures within the context of AIoTaS. Our work aims to bridge this gap and contribute to the growing body of knowledge in this important area.

AIoT DIMENSION

AIoT comprises three main dimensions: the data used to train the ML models, the ML models themselves, and the jobs used to provide the AIoT service.

Data

AIoT service providers need to properly manage their tenants' data in order to carry out the training process. Thus, in the context of AIoTaS, a service provider might collect data from all tenants and use these big data to train the model, which would be categorized as a *big dimension* of data.

However, such an approach raises important privacy concerns. In certain situations, tenants may be unwilling to share their data in the public domain. This then would be classified as a *small dimension* of data, possibly only one tenant's data. Consequently, service providers should also offer the means to train models specific to individual tenants using only the small data belonging to them.

ML models

ML models for AIoT can be categorized as either *public* models or *private* models. In the former case, the model is shared by several tenants and is generated by aggregating all of their data through centralized learning, or aggregating their local models through FL to produce a global model. In the case of private models, the model is reserved for the use of one tenant only.

Jobs

ML applications consist of two jobs: *learning* and *inference*. In AIoT, the learning process may be performed using either a centralized approach or a federated approach. In the case of centralized learning, training is performed on a single server using the data uploaded from the tenants. In FL, there are two types of jobs: training of local models and aggregation. In particular, the FL process generates multiple local (or private) models using only the data of the corresponding tenant, and these local models are then aggregated to produce a global model, which is later used to update the local models.[22] Furthermore, to perform inference in AIoT, it can be done at fog, edge, or cloud by using the global model.

SERVICES ON AIoT

AIoTaS can be implemented in three basic services: small-private, big-public, and small-public services offer different approaches to data ownership, model

training, and the tradeoff between privacy and accuracy.

Small private

In the small-private service, the models are trained using the "small" data belonging to a single tenant and the model is reserved for the use of that tenant only. This service model grants tenants exclusive control over their AI model, reducing the risks associated with data exposure and unauthorized access. The rationale behind the small-private service lies in offering a solution that prioritizes privacy concerns and empowers tenants to maintain full ownership of their data and models.

Big public

In the big-public service, the model is trained using the "big" data acquired

TABLE 1. Related works on AIoT systems.

		Data		Model		Job			Architecture				
Paper	Category	S	B	Pr	Pb	CL	FL	I	D	F	E	C	Purpose
Liu et al.[7]	ML model	—	✓	—	✓	—	✓	✓	—	—	—	✓	Efficient and accurate FL model for AIoT applications
Zhang et al.[8]		—	✓	—	✓	—	✓	✓	✓	—	—	✓	Efficient FL for IoT–cloud collaborations
Liu et al.[9]		—	✓	—	✓	—	✓	✓	—	—	—	✓	Robust FL for AIoT with malicious model detection
Dong et al.[10]		—	✓	—	✓	—	✓	✓	✓	—	✓	✓	Faster the convergence time of FL for AIoT with an adaptive privacy parameter
Liu et al.[11]		—	✓	—	✓	—	✓	✓	✓	—	✓	✓	Improve the communication efficiency of FL in edge–cloud for AIoT
Chen and Liu[12]	Architecture optimization a implementation	—	—	—	✓	✓	—	—	—	✓	✓	—	Scalable architecture for AIoT with microservices
Hao et al.[13]		—	—	—	✓	✓	—	—	—	—	✓	✓	Offloading optimization to minimize delay in AIoT
Zhu et al.[14]		—	—	—	✓	✓	—	—	—	—	✓	✓	Optimizing task scheduling in AIoT to minimize energy consumption
Raj et al.[15]		—	—	—	✓	✓	—	—	—	—	✓	✓	Framework for a fully automated and scalable process in managing AIoT data in edge
Ramos et al.[16]		—	—	—	✓	✓	—	—	—	—	✓	✓	Framework for intelligence orchestration in AIoT
Rong et al.[17]		—	—	—	✓	✓	—	—	—	—	✓	✓	Platform for deploying an AIoT application to edge–cloud efficiently
Lei et al.[18]	Survey paper	—	—	—	✓	✓	—	—	—	✓	✓	✓	Comprehensive survey on deep learning for AIoT
Chang et al.[19]		—	—	—	✓	✓	✓	—	—	✓	✓	✓	Survey on the recent approaches and technologies for supporting AIoT environment
Jiang et al.[20]		—	—	—	✓	✓	✓	—	—	—	✓	✓	Survey on collaboration modes in edge–cloud intelligence for AIoT
Zhang and Tao[21]		—	—	—	✓	✓	✓	✓	✓	✓	✓	✓	Survey of the progress, challenges, and opportunities in AIoT
Ours	Framework	✓	✓	✓	✓	✓	✓	✓	—	✓	✓	✓	AIoT job mapping to multitier architecture

B: big; C: cloud; CL: centralized learning; D: device; E: edge; F: fog; I: Inference; Pb: public; Pr: private; S: small.

from multiple tenants and is subsequently available to all of the tenants for their own use. In this service, all tenants will share their data to be trained together. Through such a service, the tenants gain access to a model with greater knowledge and accuracy, but at the expense of revealing their data in the public domain. The rationale behind the big-public service lies in leveraging the collective intelligence of a community of tenants, providing access to a high-quality model that can deliver superior performance and generalization.

Small public

The rationale behind the small-public service model combines the benefits of privacy and collaboration. Tenants who prioritize data privacy can still benefit from the knowledge and accuracy of a global model. By sharing the parameters of their local models, tenants can collectively construct a global model without revealing the specifics of their individual datasets. This rationale allows tenants to retain control over their data while contributing to the development of a more accurate and powerful model. The small-public service strikes a balance by enabling tenants to enjoy the advantages of collaborative learning and a global model without compromising the privacy and security of their sensitive data.

MULTITIER ARCHITECTURES

Previously, service providers primarily used centralized cloud computing architectures, which offered limited support for delay-sensitive services. With the advent of 5G, there's growing interest in distributed multitier architectures incorporating cloud, edge, and fog computing paradigms.

Cloud–edge–fog computing

Cloud computing provides powerful computing and storage capabilities through the use of centralized resources. However, the cloud is far from the end users, and hence significant delays are introduced, which may be unacceptable for delay-sensitive applications, such as IoT.[23]

Edge computing was introduced by ETSI under the name of *multiaccess edge computing* with the aim of virtualizing cloud capabilities into mobile network providers. Edge computing offers the same services as the cloud but with less computing capacity and is managed by mobile network operators. Notably, the edge servers can be deployed and collocated with a base station.

Fog computing was introduced by the OpenFog consortium. Fog nodes can be located anywhere between the cloud and the end devices, making them potentially the closest facility to the users. Unlike edge, which are deployed by mobile network operators, fog computing can be deployed by either individuals or businesses. Fog platforms typically consist of multiple heterogeneous nano servers, such as mobile users, vehicular fogs, and road side units.

Federation

AIoT requires significant computational power and storage capacity to process the massive amounts of data. A single computing paradigm is insufficient to meet this demand, since the capacity and coverage capabilities of the cloud, edge, and fog paradigms are all limited to a certain extent. Thus, to meet the needs of AIoT, some form of federation is needed to satisfy tenant demands. By federating these computing paradigms and establishing a multitier architecture based on a cloud–edge–fog hierarchy as defined in 5G-CORAL architecture,[24] a flexible and powerful approach for handling a wide variety of services with different characteristics is enabled.[25,26]

SERVICE-ARCHITECTURE MAPPING

This section presents the structured framework developed for guiding AIoTaS service providers in determining where to allocate each job, where to perform FL, and how many ML models to construct when implementing the services on multitier architectures.

In mapping the ML tasks to the cloud–edge–fog paradigms, the framework considers the use of a hierarchical FL approach with one-, two-, or three-federations, respectively. In the case of one-federation FL, FL is performed only once after creating the local models. In contrast, in two-federation FL, FL is performed first to aggregate the parameters of the local model to create partial-global models and then once again to aggregate these partial-global models to create a global model. Finally, in three-federation FL, FL is performed twice to generate partial-global models and is then performed a third time to create a global model. Previous studies have shown that hierarchical FL provides significant benefits in terms of a shorter model training time and lower energy usage than conventional FL.[27]

The mapping process results in 31 possible service-architecture assignments, as shown in Table 2, where these assignments are organized by the number of federations, the tiers at which FL is performed, and the number of ML models generated by each FL. It is noted that the mappings relate only to the training task since the inference task can be performed in either the cloud, the edge, or the fog.

As shown in Table 2, the entries are expressed as a string of symbols. "S"

and "B" denote small data and big data, respectively, while "C," "E," "F," "M," "N," and "K" denote cloud, edge, fog, number of edge nodes, number of fogs under each edge, and number of tenants, respectively. The backslash symbol ("\") denotes the federation of ML models belonging to the same tenant, that is, FL is performed on the local models of a single tenant. In contrast, the forward slash symbol ("/") denotes federation among different tenants, that is, FL is performed on the local models belonging to multiple tenants. Each mapping is individually annotated for ease of identification. For example, the notation SF\E\C/C indicates that small (S) data are learned into private models in the fog (F), then models of the same tenant are then federated (\) in the edge (E) and then in the cloud (C), and finally federated once again among all tenants (/) to produce a global model.

Small-private services

Small-private services can utilize either centralized or FL, with FL being performed in various tiers.

Centralized. Centralized learning is performed in the cloud. The tenant data are sent directly to the cloud and are used to generate K models (one model per tenant), with SC as the abbreviation in Table 2.

One-federation. Small-private services with one federation can be performed at the fog–cloud, edge, or edge–cloud. When utilizing the fog–cloud, local training is performed in the fog to generate local models for K tenants at MN fog nodes. All of the local models for each tenant are then

TABLE 2. Service-architecture mapping results.

	Centralized		One-federation		Two-federation		Three-federation	
Service	**Dimension**	**No. of models**	**Dimension**	**No. of models**	**Dimension**	**No. of models**	**Dimension**	**No. of models**
Small private	SC	K	SF\C	$MNK \rightarrow K$	SF\F\E	$MNK \rightarrow MK \rightarrow K$		
			SE\E	$MK \rightarrow K$	SF\E\E	$MNK \rightarrow MK \rightarrow K$		
			SE\C	$MK \rightarrow K$	SF\E\C	$MNK \rightarrow MK \rightarrow K$		
Big public	BC	1	BF\C	$MN \rightarrow 1$	BF\F\E	$MN \rightarrow M \rightarrow 1$		
			BE\E	$M \rightarrow 1$	BF\E\E	$MN \rightarrow M \rightarrow 1$		
			BE\C	$M \rightarrow 1$	BF\E\C	$MN \rightarrow M \rightarrow 1$		
Small public			SC/C	$K \rightarrow 1$	SF/F\C	$MNK \rightarrow MN \rightarrow 1$	SF\F\E/E	$MNK \rightarrow MK \rightarrow K \rightarrow 1$
					SF\C/C	$MNK \rightarrow K \rightarrow 1$	SF\F/E\E	$MNK \rightarrow MK \rightarrow M \rightarrow 1$
					SF/C\C	$MNK \rightarrow MN \rightarrow 1$	SF/F\E\E	$MNK \rightarrow MN \rightarrow M \rightarrow 1$
					SE\E/E	$MK \rightarrow K \rightarrow 1$	SF\E\E/E	$MNK \rightarrow MK \rightarrow K \rightarrow 1$
					SE/E\E	$MK \rightarrow M \rightarrow 1$	SF\E/E\E	$MNK \rightarrow MK \rightarrow M \rightarrow 1$
					SE\C/C	$MK \rightarrow K \rightarrow 1$	SF/E\E\E	$MNK \rightarrow MN \rightarrow M \rightarrow 1$
					SE/C\C	$MK \rightarrow M \rightarrow 1$	SF\E\C/C	$MNK \rightarrow MK \rightarrow K \rightarrow 1$
							SF\E/C\C	$MNK \rightarrow MK \rightarrow M \rightarrow 1$
							SF/E\C\C	$MNK \rightarrow MN \rightarrow M \rightarrow 1$

B: big data; C: cloud; E: edge; F: fog; K: number of tenants; M: number of edge nodes; N: number of fogs under each edge; and S: small data. The backslash symbol ("\") denotes the federation of ML models belonging to the same tenant; the forward slash symbol ("/") denotes federation among different tenants.

sent to the cloud for federation, generating one private model for each tenant (abbreviation SF\C).

When utilizing only the edge, tenant data are sent there to create local models for K tenants at M edge nodes. These local models are aggregated at the same tier into K private models (SE\E). In this one-tier architecture, the server that aggregates all local models is at an edge node. Similarly, local model aggregation can occur in the cloud (SE\C).

Two-federation. Small-private services can also be performed with two federations. When utilizing the fog–edge, local training is performed in the fog. Each fog under the same edge then first performs FL to aggregate MNK local models, resulting in MK partial-private models. All of the partial-private models belonging to the same tenant are then forwarded to the edge for a second FL, resulting in K private models (abbreviation SF\F\E).

In the arrangements above, various options are available for FL. For example, the first FL might alternatively be performed at the edge (abbreviation SF\E\E). Similarly, the second FL might be performed in the cloud (abbreviation SF\E\C).

Big-public services

Big-public services can utilize either centralized or FL, with federations performed in various tiers.

Centralized. The data from all of the tenants is sent directly to the cloud to perform centralized learning and generate a public model (abbreviation BC).

One- and two-Federation. Big-public services with one- and two-federation can be configured in the same manner as small-private services with the same federation. However, the two services differ in terms of the data collection method and ML model. In particular, in big-public services, the tenants share their data to the service provider for training. As a result, just one ML model, referred to hereafter as a *partial-global model*, is generated at each tier node. In the final federation, all of these partial-global models are aggregated to form a global model for all tenants. For example, in the BF\C service, the data from all of the tenants at a fog node are used to perform local training, resulting in one local model for each fog node. The local models belonging to the different fog nodes are then aggregated in the cloud to generate a global model.

Small-public services

In small-public services, an ML model is trained for each tenant using their own data, and the models are then aggregated to produce a global model for use by all of the tenants. Notably, the federation process may either be performed within the same tenant (\) or among different tenants (/).

One-federation. In one-federation, a local model is produced for each tenant in the cloud, resulting in the generation of K local models. The local models from all tenants are then aggregated to create a global model (abbreviation SC/C).

Two-federation. In two-federation, small-public services may utilize various tiers. When utilizing the fog–cloud, a local model is trained for each tenant in the fog, and first FL is performed on the local models from all tenants at each fog node to produce MN partial-global models. The partial-global models at the different fog nodes are then sent to the cloud for a second FL, resulting in a global model (abbreviation SF/F\C). Alternatively, the first FL can be performed in the cloud with a similar configuration to that of the SF/F\C service, with the exception that both federations are performed in the cloud (abbreviation SF/C\C). Finally, the first FL for each tenant may also be performed directly in the cloud, resulting in the K partial-global models. These K partial-global models are then further aggregated in the cloud to generate a public model (abbreviation SF\C/C).

When using the edge for training, tenants initially send their data to the edge to create a local model. These local models are then aggregated into K partial-global models and further combined into a global model (SE\E/E). Alternatively, an initial FL may aggregate local models from multiple tenants at each edge, producing M partial-global models. A second FL process then generates a global model by aggregating these partial-global models (SE/E\E).

When employing the edge–cloud for training, the configurations are similar to the mappings that utilize fog–cloud. For example, SE\C/C and SE/C\C configurations are similar to SF\C/C and SF/C\C, respectively, with the exception that the local model is trained at edge nodes instead of fog nodes.

Three-federation. In three-federation, small-public services perform local training in the fog. First, FL may be performed in either the fog or the edge, while the second and third FL may be performed in the edge or the cloud.

In one case, FL in fog generates MK partial-global models from tenant

local models. Then, either the models are aggregated into *K* partial-global models in a second FL and finally into a global model in a third FL (SF\F\E/E), or a second FL aggregates local models at each edge into *M* partial-global models, which are then aggregated into a global model in a third FL (SF\F/E\E).

First, an FL may also be performed among all tenants at each fog node, aggregating local models to yield *MN* partial-global models. Second, an FL then combines these models under the same edge node. Finally, a third FL aggregates public models across different edge nodes to form a global model (denoted as SF/F\E\E).

The mapping described previously is also close to the case where the second FL is carried at the edge. For example, the mappings of SF\E\E/E, SF\E/E\E, and SF/E\E\E are similar to SF\F\E/E, SF\F/E\E, and SF/F\E\E, respectively. On the other hand, the mappings differ in terms of the tier location for carrying out the second FL.

For the initial and second FL performed in the edge and cloud, each tenant's local models first aggregate at the edge, creating *MK* partial-global models. The SF\E\E/E and SF\E/E\E processes are similar, but both FL stages happen in the cloud instead (SF\E\C/C and SF\E/C\C). Alternatively, edge FL could involve all tenants, with second and third FLs taking place in the cloud (SF/E\C\C).

Inference

The inference task can be performed in the cloud, edge, or fog. However, if the inference task and final FL are handled by different tiers, the model obtained from the final FL must be delivered to the tier assigned to perform inference. For example, if the inference task is performed on the edge in the SF\C/C, the global model generated in the cloud must be transferred to the fog and edge to update the local models and inference models, respectively.

NOTABLY, SUCH SMALL-PUBLIC CONFIGURATIONS RETAIN THE ADVANTAGE OF SMALL-PRIVATE CONFIGURATIONS IN THAT THE ORIGINAL DATA ARE STILL KEPT PRIVATE.

Mapping examples

Figure 2 illustrates three typical mapping results for small-private, big-public, and small-public services, in which local training and first FL are performed in the fog and edge, respectively; In every case, the second and third FLs are performed either in the edge or in the cloud.

Figure 3 presents a detailed schematic of the BF\E\C system operation flow for illustration purposes, with the inference process assigned to the fog. The training data are sent to the fog by the IoT devices, and local training is performed to generate a local model for each fog node. Each local model has a set of model parameters for saving the model characteristics. The model parameters are then sent to the edge for first FL with the model parameters of the other fog nodes. After the first FL, the new model parameters are pushed back to all of the fogs to update the respective local models. The model parameters are additionally forwarded to the cloud for the second FL to generate a global model. Finally, the global model is pushed back to all the fogs to update the local models.

Service comparisons

In providing AIoTaS, the operational costs, including computation and communication, are a major concern for service providers. *Computing cost* refers to the cost of operating the running servers, whereas *communication cost* refers to the cost of transmitting data from user equipment (UE) to servers. To offer service providers a better understanding of the services in AIoTaS, we compare the costs of all 31 mappings in Figure 4. We then discuss the tradeoff between these computing and communication costs when utilizing the services.

Figure 4(a) shows the comparisons for small private and big public. The most recommended way to run those two services with a balanced cost is to use only edge. Edge computing has lower operational costs than fog. The more distributed the servers, the higher the corresponding costs for cooling systems, maintenance staff, and other applications. Furthermore, delivering data to the edge network is also cheap because the link is close to the UEs. Using the cloud, on the other hand, is expensive because all of the data have to travel a great distance. Such a distance is particularly unsuitable for some AIoT applications that demand real-time response, as it will

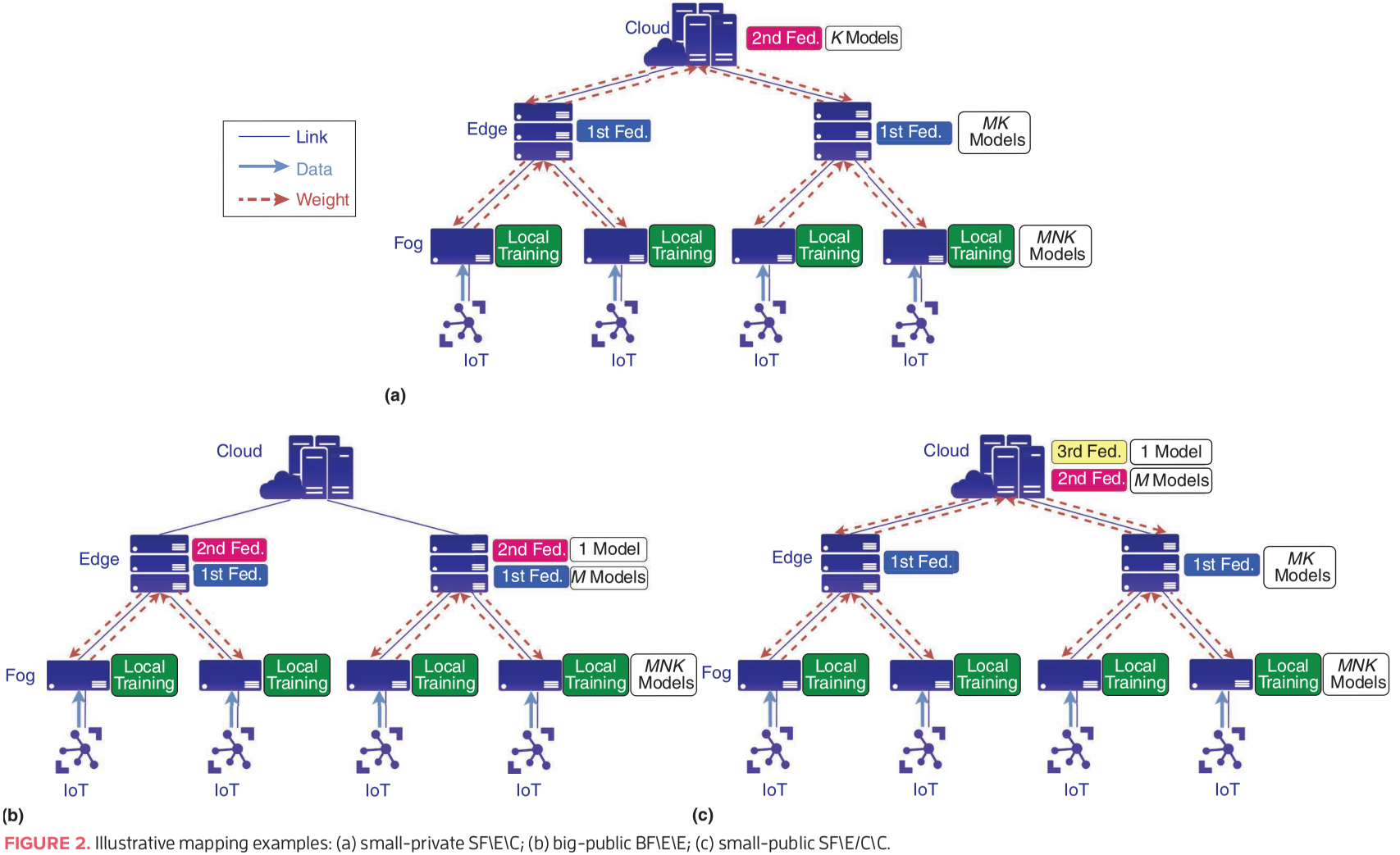

FIGURE 2. Illustrative mapping examples: (a) small-private SF\E\C; (b) big-public BF\E\E; (c) small-public SF\E/C\C.

take up to 2 s to transmit the data to cloud, delaying the decision-making process. Using fog as the closest thing to the UEs, on the other hand, results in higher operational costs than using the edge.

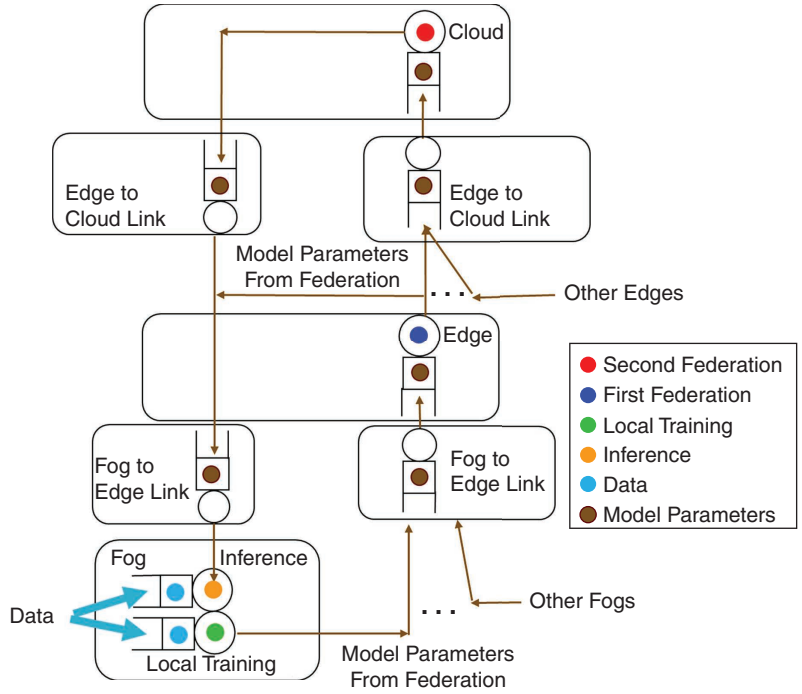

FIGURE 3. Operational flow of illustrative BF\E\C system.

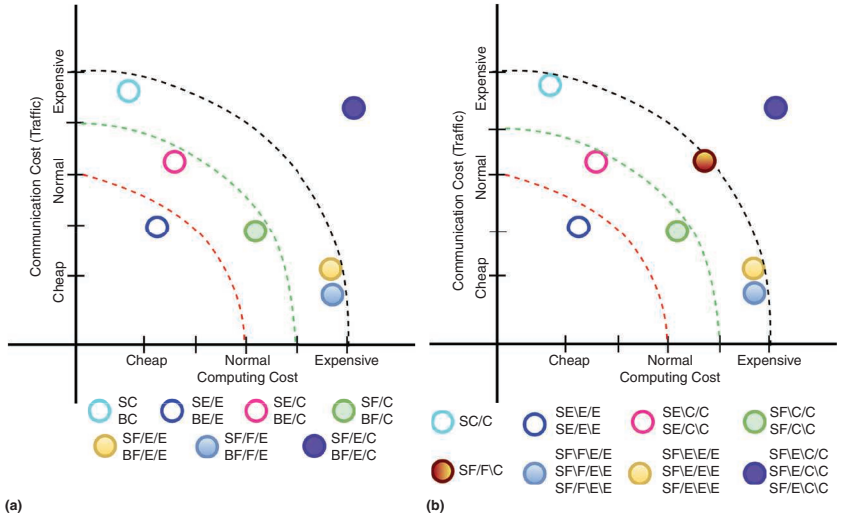

FIGURE 4. (a) and (b) Services comparisons.

Figure 4(b) shows the comparisons of small-public services. Similarly to the previous discussion, in providing small-public mode, using just edge computing, such as SE\E/E or SE/E\E mapping, is the most preferred configuration. Moreover, using fog–cloud or edge–cloud federation can be another option. Although those federations are more expensive, they offer a more scalable and flexible method for accommodating tenants' needs.

Our comparisons have been validated by the findings in Lai et al.,[28] where extensive experiments were conducted to identify the best multitier architecture configurations in providing the services. While the services employed differ from our work, the underlying principles of the optimization approaches and results can be applicable and adopted in our context. The research clearly demonstrated that allocating all tasks to the edge resulted in the best performance among the evaluated configurations. This outcome supports our analysis, highlighting the benefits of leveraging edge computing for optimizing AIoTaS services. By referencing these validated results, we enhance the credibility of our discussion and provide a robust foundation for our AIoTaS service comparison.

OPEN CHALLENGES

Although AIoTaS provides many exciting opportunities for tenants and service providers, several key challenges must be addressed before it can be commercialized.

Architecture optimization

Architecture optimization is an essential activity aimed at minimizing the resources required while satisfying the AIoT latency constraints. As described previously, a total of 31

possible services with the comparisons of each service have been identified. However, further optimization studies are still required to determine the optimal services, which have the minimum delay and/or capacity.

Security

When implementing AIoTaS, the network and architecture have to be protected. For example, when utilizing centralized learning, it is essential to protect the data as it is transmitted from the tenant(s) to the server. While the data can be protected through encryption, this increases both the computational burden on the system and the delay. Thus, alternative methods for securing the data transmissions must be found. Furthermore, it is also necessary to protect the ML algorithm. In practice, ML models are vulnerable to a wide range of attacks. For example, a malicious third party may seek to attack the FL process by sending adversarial local models to the aggregation server, thereby increasing the convergence time and degrading the model performance.

AI-based next-generation computing for AIoTaS

The articles by Iftikhar et al.[29] and Gill et al.[30] provide valuable insights into emerging trends and future directions in AI for next-generation computing. Expanding upon the insights from these articles, we deepen the conversation on AI-based next-generation computing for AIoT as a service. We specifically examine future paths and research challenges, enriching the discussion on previously addressed topics.

AI for AIoTaS. The integration of AI with IoT technologies presents a transformative opportunity to enhance the functionality, efficiency, and intelligence of IoT systems. The development of AI algorithms specifically designed for IoT applications is an area of significant interest and importance in the field.

Currently, combining FL with reinforcement learning (RL) can be a promising approach to improving the quality of AI models for AIoTaS. FL can be employed to train a global model using data from multiple IoT devices, while RL can be used to optimize and fine-tune the global model based on the specific requirements and objectives of the AIoT service.

This approach combines the benefits of diverse data sources and adaptive learning. It leverages distributed devices for a more comprehensive dataset, boosting model accuracy. RL further refines the global model through continuous feedback, enhancing adaptability to dynamic conditions in AIoT service.

However, this combination presents exciting research directions and challenges, as outlined below.

- *Efficient model aggregation:* Developing more efficient algorithms for aggregating FL and RL models from distributed architectures in AIoT environments. This includes exploring techniques to minimize communication overhead, improve convergence speed, and handle heterogeneity across devices.
- *Exploration–exploitation tradeoffs:* Balancing exploration and exploitation in RL within the FL framework, considering that different edge devices may have varying levels of exploration capability; developing algorithms that can adaptively balance exploration and exploitation to improve the overall performance and convergence of the FL–RL models.
- *System-level optimization:* Considering the system-level optimization of FL–RL in AIoTaS. This includes optimizing the allocation of computational resources, communication bandwidth, and energy consumption across multiple fog–edge–cloud nodes to maximize the overall performance and efficiency of the AIoT system.

Intelligent service framework management for AIoTaS. A critical aspect of delivering efficient and effective AIoTaS is the development of an intelligent service management framework that acts as an orchestrator, managing the federation of cloud, edge, and fog devices, and proxy as the helper to manage the nodes for orchestrator. Figure 5 shows the illustration of the service framework with the orchestrator that may be used to manage and control the whole system.

Equipped with an intelligent autodevelopment feature, the framework utilizes AI to seamlessly deploy the AIoT service from a tenant to the distributed architectures, including cloud, edge, and fog. AI helps in the automatic scheduling of such large-scale AIoT systems, taking into account factors such as resource availability, latency requirements, and workload characteristics to optimize the deployment process.

The framework also incorporates intelligent autoscaling, using AI to dynamically adjust resources based on real-time demand. Traditional autoscaling methods rely on predefined thresholds or simple prediction models, which may not adapt quickly enough to sudden changes in demand or efficiently

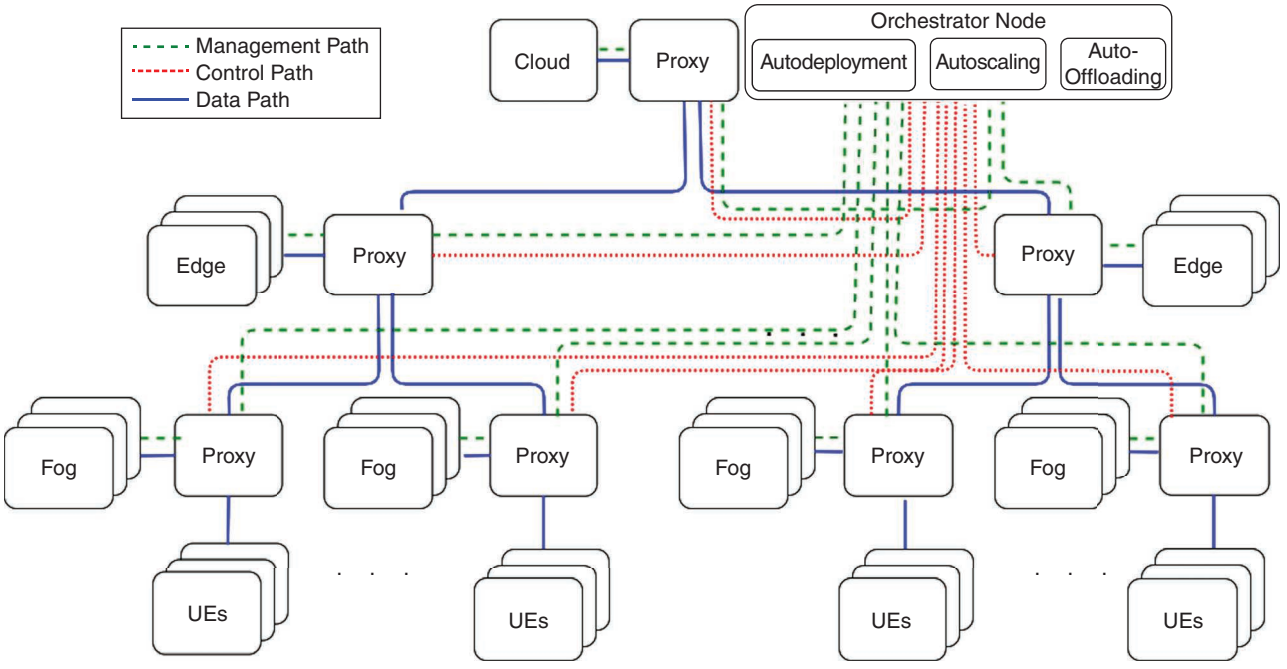

FIGURE 5. Illustrative service framework architecture.

handle the complex dependencies between different resources. However, AI-driven autoscaling employs more sophisticated prediction models that can adapt to varying demand patterns and manage resource interdependencies, improving system responsiveness and resource efficiency.

Furthermore, the framework includes intelligent auto-offloading, where AI plays a crucial role in determining how and when to offload tasks from one device or layer (for example, edge) to another (for example, cloud) based on factors such as task characteristics, network conditions, and device capacities. This helps in minimizing latency, reducing bandwidth usage, and balancing load across devices.

Despite these advantages, there are several future directions and challenges to realize such a service framework. Developing AI models that can effectively handle the complexity and dynamics of large-scale, heterogeneous AIoT systems is a significant challenge. Finally, ensuring the robustness and reliability of AI-driven decisions in the face of uncertainties and anomalies in real-world AIoT environments is another crucial challenge.

Quantum computing for AIoTaS. As highlighted by Gill et al.,[30] quantum computing holds the potential to fundamentally reshape AI. Similarly, it could significantly impact the realm of AIoT as a service system, especially in contexts demanding large-scale computation and data analysis. It could speed up certain calculations, including complex optimization and ML tasks, thus potentially enhancing the speed, efficiency, and capabilities of AI algorithms in AIoT systems. Moreover, the large volume of data generated by IoT devices could potentially be processed more efficiently using quantum computers, given their capacity for parallel computation and superior computational power.

However, despite its potential, quantum computing is still in its early stages of development and faces significant technical challenges, as outlined below.

- *Technical difficulties:* One of the key challenges lies in overcoming technical issues related to quantum bit (qubit) stability and error correction. Quantum computing still needs significant advancements to ensure the consistent and error-free operation of quantum computers.
- *Integration with existing technologies:* The task of integrating quantum computing with the existing technologies utilized in AIoTS systems remains a

considerable challenge and an active area of research. The development of a robust framework capable of accommodating these varied technologies presents significant difficulties.

- *Applicability in AIoT systems:* Not all AIoT systems will require the computational power offered by quantum computing. For many IoT applications, traditional computing technologies may still be sufficient, when combined with techniques such as fog or edge computing to reduce data transmission and cloud computing for scalable storage and processing.

This study has presented a generic framework for constructing three possible services of AIoTaS: small private, big public, and small public. By mapping the training and federation tasks to the cloud–edge–fog paradigms, 31 possible mappings have been identified, comprising seven small-private services, seven big-public services, and 17 small-public services. Therefore, it is expected that the majority of AIoTaS applications might be deployed as big-public services, with only the most sensitive applications being deployed as small-private services. In terms of operational costs, employing only the edge is preferable for running services on architecture. However, further optimization studies are required to determine the optimal mapping for each service, which minimizes the service delay to the user and the capacity cost to the provider. Furthermore, additional work is needed to develop an effective service framework for managing the operations of the distributed AIoTaS architecture.

ABOUT THE AUTHORS

YING-DAR LIN is a chair professor of computer science at National Yang Ming Chiao Tung University, Hsinchu 300, Taiwan. His research interests include network softwarization, cybersecurity, and wireless communications. Lin received a Ph.D. in computer science from the University of California at Los Angeles. He is an IEEE Distinguished Lecturer, has served or is serving on the editorial boards of several IEEE journals and magazines, and was the editor in chief of *IEEE Communications Surveys and Tutorials* from 2016 to 2020. He is an IEEE Fellow. Contact him at ydlin@cs.nctu.edu.tw.

YUAN-CHENG LAI is a distinguished professor in the Department of Information Management, National Taiwan University of Science and Technology, Taipei 106, Taiwan. His research interests include performance analysis, software-defined networking, wireless networks, and Internet of Things security. Lai received a Ph.D. from the Department of Computer and Information Science, National Chiao Tung University. Contact him at laiyc@cs.ntust.edu.tw.

DIDIK SUDYANA is a Ph.D. candidate in the Electrical Engineering and Computer Science International Graduate Program of National Yang Ming Chiao Tung University, Hsinchu 300, Taiwan, and a lecturer at Sekolah Tinggi Manajemen Informatika and Komputer AMIK Riau, 28294 Riau, Indonesia. His research interests include cybersecurity, machine learning, and network design and optimization. Sudyana received an M.S. in informatics from Universitas Islam Indonesia, Indonesia. Contact him at dsudyana@cs.nctu.edu.tw.

REN-HUNG HWANG is the dean of the College of Artificial Intelligence, National Yang Ming Chiao Tung University Hsinchu 300, Taiwan. His research interests include deep learning, wireless communications, network security, and cloud–edge–fog computing. Hwang received a Ph.D. in computer science from the University of Massachusetts, Amherst. He is an IEEE Fellow. Contact him at rhhwang@nycu.edu.tw.

REFERENCES

1. W. Wei, "Guest editorial: Special section on integration of big data and artificial intelligence for internet of things," *IEEE Trans. Ind. Informat.*, vol. 16, no. 4, pp. 2562–2565, Apr. 2020, doi: 10.1109/TII.2019.2958638.
2. B. K. Kuguoglu, "The giant leap for smart cities: Scaling up smart city artificial intelligence of things (AIoT) initiatives," *Sustainability*, vol. 13, no. 21, 2021, Art. no. 12295, doi: 10.3390/su132112295.
3. G. Drainakis, K. V. Katsaros, P. Pantazopoulos, V. Sourlas, and

A. Amditis, "Federated vs. centralized machine learning under privacy-elastic users: A comparative analysis," in *Proc. IEEE 19th Int. Symp. Netw. Comput. Appl. (NCA)*, 2020, pp. 1–8, doi: 10.1109/NCA51143.2020.9306745.

4. A. E. Ouadrhiri and A. Abdelhadi, "Differential privacy for deep and federated learning: A survey," *IEEE Access*, vol. 10, pp. 22,359–22,380, Feb. 2022, doi: 10.1109/ACCESS.2022.3151670.
5. F. Firouzi, B. Farahani, and A. Marinšek, "The convergence and interplay of edge, fog, and cloud in the AI-driven Internet of Things (IoT)," *Inf. Syst.*, vol. 107, Jul. 2022, Art. no. 101840, doi: 10.1016/j.is.2021.101840.
6. B. Kar, W. Yahya, Y.-D. Lin, and A. Ali, "Offloading using traditional optimization and machine learning in federated cloud–edge–fog systems: A survey," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 2, pp. 1199–1226, Secondquarter 2023, doi: 10.1109/COMST.2023.3239579.
7. T. Liu, J. Xia, Z. Ling, X. Fu, S. Yu, and M. Chen, "Efficient federated learning for AIoT applications using knowledge distillation," *IEEE Internet Things J.*, vol. 10, no. 8, pp. 7229–7243, Apr. 2023, doi: 10.1109/JIOT.2022.3229374.
8. X. Zhang, M. Hu, J. Xia, T. Wei, M. Chen, and S. Hu, "Efficient federated learning for cloud-based AIoT applications," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 11, pp. 2211–2223, Nov. 2021, doi: 10.1109/TCAD.2020.3046665.
9. W. Liu et al., "D2MIF: A malicious model detection mechanism for federated-learning-empowered artificial intelligence of things," *IEEE Internet Things J.*, vol. 10, no. 3, pp. 2141–2151, Feb. 2023, doi: 10.1109/JIOT.2021.3081606.
10. F. Dong et al., "PADP-FedMeta: A personalized and adaptive differentially private federated meta learning mechanism for AIoT," *J. Syst. Archit.*, vol. 134, Jan. 2023, Art. no. 102754, doi: 10.1016/j.sysarc.2022.102754.
11. Y. Liu, P. Huang, F. Yang, K. Huang, and L. Shu, "QuAsyncFL: Asynchronous federated learning with quantization for cloud-edge-terminal collaboration enabled AIoT," *IEEE Internet Things J.*, early access, 2023, doi: 10.1109/JIOT.2023.3290818.
12. C.-H. Chen and C.-T. Liu, "A 3.5-tier container-based edge computing architecture," *Comput. Electr. Eng.*, vol. 93, Jul. 2021, Art. no. 107227, doi: 10.1016/j.compeleceng.2021.107227.
13. Y. Hao, Y. Miao, L. Hu, M. S. Hossain, G. Muhammad, and S. U. Amin, "Smart-edge-CoCaCo: AI-enabled smart edge with joint computation, caching, and communication in heterogeneous IoT," *IEEE Netw.*, vol. 33, no. 2, pp. 58–64, Mar./Apr. 2019, doi: 10.1109/MNET.2019.1800235.
14. S. Zhu, K. Ota, and M. Dong, "Energy-efficient artificial intelligence of things with intelligent edge," *IEEE Internet Things J.*, vol. 9, no. 10, pp. 7525–7532, May 2022, doi: 10.1109/JIOT.2022.3143722.
15. E. Raj, D. Buffoni, M. Westerlund, and K. Ahola, "Edge MLOps: An automation framework for AIoT applications," in *Proc. IEEE Int. Conf. Cloud Eng. (IC2E)*, 2021, pp. 191–200, doi: 10.1109/IC2E52221.2021.00034.
16. E. J. Ramos, M.-J. Montpetit, A. F. Skarmeta, M. Boussard, V. Angelakis, and D. Kutscher, "Architecture framework for intelligence orchestration in AIoT and IoT," in *Proc. Int. Conf. Smart Appl., Commun. Netw. (SmartNets)*, 2022, pp. 1–4, doi: 10.1109/SmartNets55823.2022.9994029.
17. G. Rong, Y. Xu, X. Tong, and H. Fan, "An edge-cloud collaborative computing platform for building AIoT applications efficiently," *J. Cloud Comput., Adv., Syst. Appl.*, vol. 10, no. 1, Jul. 2021, Art. no. 36, doi: 10.1186/s13677-021-00250-w.
18. L. Lei, Y. Tan, K. Zheng, S. Liu, K. Zhang, and X. Shen, "Deep reinforcement learning for autonomous internet of things: Model, applications and challenges," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 1722–1760, thirdquarter 2020, doi: 10.1109/COMST.2020.2988367.
19. Z. Chang, S. Liu, X. Xiong, Z. Cai, and G. Tu, "A survey of recent advances in edge-computing-powered artificial intelligence of things," *IEEE Internet Things J.*, vol. 8, no. 18, pp. 13,849–13,875, Sep. 2021, doi: 10.1109/JIOT.2021.3088875.
20. K. Jiang, C. Sun, H. Zhou, X. Li, M. Dong, and V. C. M. Leung, "Intelligence-empowered mobile edge computing: Framework, issues, implementation, and outlook," *IEEE Netw.*, vol. 35, no. 5, pp. 74–82, Sep./Oct. 2021, doi: 10.1109/MNET.101.2100054.
21. J. Zhang and D. Tao, "Empowering things with intelligence: A survey of the progress, challenges, and opportunities in artificial intelligence of things," *IEEE Internet Things J.*, vol. 8, no. 10, pp. 7789–7817, May 2021, doi: 10.1109/JIOT.2020.3039359.
22. P. Kairouz et al., "Advances and open problems in federated learning," 2021, *arXiv:1912.04977*.
23. J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative cloud and edge computing for latency minimization," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 5031–5044, May 2019, doi: 10.1109/TVT.2019.2904244.

24. P.-H. Kuo et al., "An integrated edge and fog system for future communication networks," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshops (WCNCW)*, 2018, pp. 338–343, doi: 10.1109/WCNCW.2018.8369023.
25. A. Yousefpour et al., "All one needs to know about fog computing and related edge computing paradigms: A complete survey," *J. Syst. Archit.*, vol. 98, pp. 289–330, Sep. 2019, doi: 10.1016/j.sysarc.2019.02.009.
26. G. Merlino, "Enabling workload engineering in edge, fog, and cloud computing through OpenStack-based middleware," *ACM Trans. Internet Technol.*, vol. 19, no. 2, pp. 1–22, May 2019, doi: 10.1145/3309705.
27. L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2020, pp. 1–6, doi: 10.1109/ICC40277.2020.9148862.
28. Y.-C. Lai et al., "Task assignment and capacity allocation for ml-based intrusion detection as a service in a multi-tier architecture," *IEEE Trans. Netw. Service Manag.*, vol. 20, no. 1, pp. 672–683, Mar. 2023, doi: 10.1109/TNSM.2022.3203427.
29. S. Iftikhar et al., "AI-based fog and edge computing: A systematic review, taxonomy and future directions," *Internet Things*, vol. 21, Apr. 2023, Art. no. 100674, doi: 10.1016/j.iot.2022.100674.
30. S. S. Gill et al., "AI for next generation computing: Emerging trends and future directions," *Internet Things*, vol. 19, Aug. 2022, Art. no. 100514, doi: 10.1016/j.iot.2022.100514.

Digital Object Identifier 10.1109/MC.2023.3326557