AIoT as a Service: Small or Big Data, Private or Public Model, Centralized or Federated Learning?

Journal:	*IEEE Internet of Things Magazine*
Manuscript ID	IOTMAG-22-00141
Topic or Series:	General Open Call for Articles
Date Submitted by the Author:	29-Jun-2022
Complete List of Authors:	Lin, Ying-Dar; National Chiao Tung University, Computer Science Lai, Yuan-Cheng; National Taiwan University of Science and Technology, Information Management Sudyana, Didik; National Yang Ming Chiao Tung University, Computer Science Hwang, Ren-Hung; National Chung Cheng University, Computer Science Information Engineering
Key Words:	AIoT as a service, big data, small data, federated learning, cloud, edge, fog, federation

AIoT as a Service: Small or Big Data, Private or Public Model, Centralized or Federated Learning?

Ying-Dar Lin, *Fellow, IEEE*, Yuan-Cheng Lai, Didik Sudyana, Ren-Hung Hwang, *Senior Member, IEEE*

***Abstract*—Internet of Things (IoT) pumps large amount of data that could be better recognized by Artificial Intelligence (AI). This form of AI for IoT (AIoT) could be constructed by IoT owners themselves, or outsourced to service providers in a shared form of AIoT called AIoT as a Service (AIoTaS). In the latter case, cloud and edge service providers, with global coverage and low latency, respectively, compete for the market of AIoTaS. They could also federate into cloud-edge-fog paradigms to provide a full spectrum of services. For example, IoT owners, i.e., AIoTaS tenants, can share their data as a big data to train a public model, known as a big-data-public-model (or big-public in short). Alternatively, those with privacy concerns can use their small data to train individual private models (i.e., small-private), or further merge private models into a public model by federated learning (i.e., small-public). Accordingly, we propose a generic framework for mapping the training and federation tasks of three AIoTaS services, namely big-public, small-private, and small-public, to multiple cloud-edge-fog paradigms. Each mapping is individually annotated for ease of identification. For example, the notation SF\E\C/C indicates that small (S) data is learned into private models in the fog (F), then federated among the same tenants (\) at the edge (E) and then the cloud (C), and finally federated once again among all tenants (/) to produce a public model. A total of 31 possible mappings are identified as possible reference designs for AIoTaS providers, including 7 for small-private, 7 for big-public, and 17 for small-public.**

***Index Terms*—AIoT as a service, big data, small data, federated learning, cloud, edge, fog, federation.**

I. Introduction

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, this 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.

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 the devices and manage them. These tasks impose a significant burden on the capabilities of the owners and developers, 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 (i.e., 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, this 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 is owned by a tenant and is used to train a private model, whereas big data is owned by all or some of the tenants and is used to train a public model. Public models have a higher accuracy than private models since they acquire a global knowledge from all the data received from the tenants [1]. However, private models have superior privacy since the data on which they are trained is not shared. Furthermore, tenants can improve the accuracy of these models by leveraging federated learning (FL) to generate public models based on the aggregated private models of different tenants. Notably, such small-public configurations retain the advantage of small-private in that the original data is still kept private.

Another important consideration when deploying AIoT platforms is the service architecture to be employed [2]. 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. Note that the concepts of federation and FL are different, where the former forms the integration at the lower platform level while the latter integrates the upper-level machine learning process. They don't need to co-exist. Nevertheless, it is possible to combine both by running FL for AIoTaS over the federated cloud-edge-fog platform.

This paper 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

Ying-Dar Lin and D. Sudyana are with the Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu 300, Taiwan (e-mail: ydlin@cs.nctu.edu.tw; dsudyana@cs.nctu.edu.tw).

Yuan-Cheng Lai is with the Department of Information Management, National Taiwan University of Science and Technology, Taipei 106, Taiwan (e-mail: laiyc@cs.ntust.edu.tw).

Ren-Hung Hwang is with the Department of Computer Science Information Engineering, National Chung Cheng University, Chiayi 60102, Taiwan, and also with the AI College, National Yang Ming Chiao Tung University, Tainan 711, Taiwan (e-mail: rhhwang@cs.ccu.edu.tw).

TABLE I: Related works on AIoT systems

Paper	Category	Data		Model		Job			Architecture				Purpose
		S	B	Pr	Pb	CL	FL	Dt	D	F	E	C	
[3]	ML Model	-	✓	-	✓	-	✓	✓	-	-	-	✓	Efficient and accurate FL model for AIoT applications
[4]		-	✓	-	✓	-	✓	✓	✓	-	-	✓	Efficient FL for IoT-Cloud collaborations
[5]		-	✓	-	✓	-	✓	✓	-	-	-	✓	Robust FL for AIoT with malicious model detection
[6]		-	✓	-	✓	-	✓	✓	-	-	✓	✓	Mitigating test error caused by erroneous training data in FL for AIoT
[7]	Architecture Optimization & Implementation	-	-	-	✓	✓	-	-	-	✓	✓	-	Scalable architecture for AIoT with microservices
[8]		-	-	-	✓	✓	-	-	-	-	✓	✓	Offloading optimization to minimize delay in AIoT
[9]		-	-	-	✓	✓	-	-	-	-	✓	✓	Framework for a fully automated and scalable process in managing AIoT data in Edge
[10]	Survey Paper	-	-	-	✓	✓	-	-	-	✓	✓	✓	Comprehensive survey on deep learning for AIoT
[11]		-	-	-	✓	✓	✓	-	-	✓	✓	✓	Survey on the recent approaches and technologies for supporting AIoT environment
[12]		-	-	-	✓	✓	✓	-	-	-	✓	✓	Survey on collaboration modes in edge-cloud intelligence for AIoT
Ours	Framework	✓	✓	✓	✓	✓	✓	✓	-	✓	✓	✓	AIoT job mapping to Multi-tier Architecture

S/B: Small/Big, Pr/Pb: Private/Public, CL/FL/Dt: Centralized Learning/Federated Learning/Detection, D/F/E/C: Device/Fog/Edge/Cloud

introduced and described, and the open challenges facing the commercialization of AIoTaS are then briefly discussed.

The remainder of this paper is organized as follows. Section II reviews previous work in the field. Section III explains the main dimensions of AIoT. Section IV describes three service modes of AIoTaS. Section V discusses the multi-tier architectures to support AIoTaS. Section VI presents the service-architecture mapping results. Section VII describes the open challenges facing the AIoTaS field. Finally, Section VIII provides some brief concluding remarks.

II. Related Works

Table 1 summarizes the previous work on AIoT systems. As shown, these studies have mainly set out to examine the specific ML models or architectures used to support AIoT, or to survey the work in the field. The studies on ML consider to increase the FL accuracy [3][4] and robustness [5], or mitigate the FL process error [6].

Previous studies on architecture optimization and implementation consider public models with centralized learning and focus mainly on issues such as scalable AIoT for microservices [7], optimization offloading for minimizing delay [8], and scalable framework development for AIoT services [9].

Finally, previous survey papers have reviewed existing ML algorithms for AIoT [10], recent approaches and technologies for the support of AIoT [11], and the use of collaboration modes in the edge and cloud to support AIoT [12].

However, none of these studies have considered AIoTaS. Moreover, the literature lacks a robust effort to examine the data, model and job dimensions of AIoT and to map these dimensions to appropriate architectures in the context of AIoTaS.

III. 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.

A. 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 using this 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.

B. 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 (CL), or aggregating their local models through federated learning (FL). In the case of private models, the model is reserved for the use of one tenant only.

C. Jobs

ML applications consist of two jobs: learning and detection. 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. By contrast, in FL, training is conducted across multiple servers using local data. 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 public model which is later used to update the local models [13].

IV. Services on AIoT

AIoTaS can be implemented in three basic service modes: small-private, big-public, and small-public.

A. Small-Private

In the small-private mode, 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. The service is thus appropriate for tenants who have privacy concerns since neither the data nor the model are shared with others.

B. Big-Public

In the big-public mode, the model is trained using the "big" data acquired from multiple tenants and is subsequently available to all of the tenants for their own use. 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.

C. Small-Public

In the small-public mode, private models are trained using the "small" data owned by individual tenants, and these models are then aggregated by the service provider to construct a public model. Through this service, tenants are able to retain the privacy of their data through not merging it with that of others, but are also able to enjoy the benefits of a public model with greater accuracy.

V. Multi-tier Architectures

In the past, many service providers relied on centralized architectures with cloud computing. However, such architectures provide only limited support to delay-sensitive services. Thus, fueled by the emergence of 5G technology, distributed multi-tier architectures based on cloud, edge, and fog computing paradigms have attracted significant interest in recent years.

A. 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.

Edge computing was introduced by ETSI under the name of Multi-Access Edge Computing (MEC) 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 (RSUs).

B. 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. Previous studies have shown that federation provides a flexible and powerful approach for handling a wide variety of services with different characteristics [14].

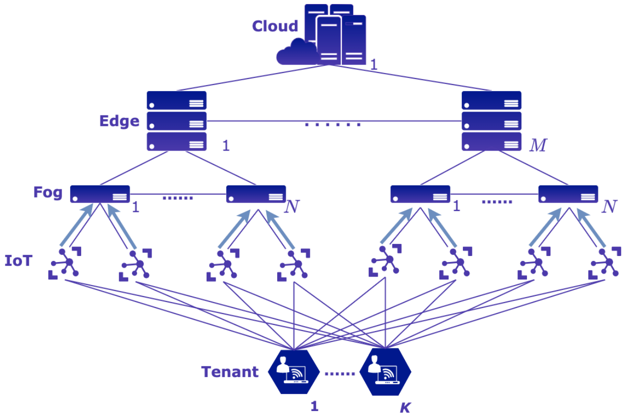

Fig. 1: Typical multi-tier architecture for AIoT system.

C. System Architecture

Figure 1 illustrates a typical multi-tier architecture composed of a cloud C, M edges E, N fogs F under each edge, and K tenants. Utilizing such an architecture, service providers can implement two-tier in cloud-fog or edge-fog, and three-tier in cloud-edge-fog.

VI. 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 multi-tier architectures.

In mapping the ML tasks to the cloud-fog-edge paradigms, the framework considers the use of a hierarchical FL approach with 1-, 2-, or 3-federations, respectively. In the case of 1-federation FL, federation is performed only once after creating the local models. By contrast, in 2-federation FL, federation is performed first to aggregate the parameters of the local model to create partial-public models and then once again to aggregate these partial-public models to create a public model. Finally, in 3-federation FL, federation is performed twice to generate partial-public models and is then performed a third time to create a public 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 [15].

The mapping process results in 31 possible service-architecture assignments, as shown in Table II, where

TABLE II: Service-architecture mapping results

Service	Centralized		1-Federation		2-Federation		3-Federation	
	Dimension	# of models	Dimension	# of models	Dimension	# of models	Dimension	# of models
Small-Private	SC	K	SF\C	MNK→K	SF\F\E	MNK→MK→K		
			SE\E	MK→K	SF\E\E	MNK→MK→K		
			SE\C	MK→K	SF\E\C	MNK→MK→K		
Big-Public	BC	1	BF\C	MN→1	BF\F\E	MN→M→1		
			BE\E	M→1	BF\E\E	MN→M→1		
			BE\C	M→1	BF\E\C	MN→M→1		
Small-Public			SC/C	K→1	SF/F\C	MNK→MN→1	SF\F\E/E	MNK→MK→K→1
					SF\C/C	MNK→K→1	SF\F/E\E	MNK→MK→M→1
					SF/C\C	MNK→MN→1	SF/F\E\E	MNK→MN→M→1
					SE\E/E	MK→K→1	SF\E\E/E	MNK→MK→K→1
					SE/E\E	MK→M→1	SF\E/E\E	MNK→MK→M→1
					SE\C/C	MK→K→1	SF/E\E\E	MNK→MN→M→1
					SE/C\C	MK→M→1	SF\E\C/C	MNK→MK→K→1
							SF\E/C\C	MNK→MK→M→1
							SF/E\C\C	MNK→MN→M→1

these assignments are organized by the number of federations, the tiers at which federation is performed, and the number of ML models generated by each federation. It is noted that the mappings relate only to the training task since the detection task can be performed in either the cloud, the edge, or the fog.

As shown, the table entries are expressed as a string of symbols. S and B denote small data and big data, respectively; while C, E, and F denote cloud, edge, and fog, respectively. The symbol ’$\backslash$’ denotes the federation of ML models belonging to the same tenant, i.e., federation is performed on the local models of a single tenant. By contrast, the symbol ’$/$’ denotes federation among different tenants, i.e., federation is performed on the local models belonging to multiple tenants.

A. Small-Private Services

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

1) 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). (Abbreviation SC in Table II.)

2) 1-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 sent to the cloud for federation, generating one private model for each tenant. (Abbreviation SF\C).

When utilizing only the edge, the tenant data are sent to the edge to generate local models for K tenants at M edge nodes. The local models belonging to the same tenant are then aggregated at the edge to generate K private models. (Abbreviation SE\E). Federation of the local models could equally be performed in the cloud. (Abbreviation SE\C).

3) 2-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 performs first federation 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 second federation, resulting in K private models. (Abbreviation SF\F\E).

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

B. Big-Public Services

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

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

2) 1- and 2-Federation: Big-public services with 1- and 2-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-public model, is generated at each tier node. In the final federation, all of these partial-public models are aggregated to form a public model for all tenants. For example, in the BF\C service, the data from all 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 federated in the cloud to generate a public model.

C. Small-Public Services

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

1) 1-Federation: In 1-federation, a local model is produced for each tenant, resulting in the generation of K local models. The local models from all tenants are then aggregated in the cloud to create a public model. (Abbreviation SC/C).

2) 2-Federation: In 2-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 federation is performed on the local models from all tenants at each fog node to produce MN partial-public models. The partial-public models at the different fog nodes are then sent to the cloud for second federation, resulting in a public model. (Abbreviation SF/F\C). Alternatively, the first federation 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 federation for each tenant may also be performed directly in the cloud, resulting in the K partial-public models. These K partial-public models are then further aggregated in the cloud to generate a public model. (Abbreviation SF\C/C).

When using the edge for training, each tenant first sends their data to the edge to generate a local model. For each tenant, the local models at other edges are aggregated to produce a total of K partial-public models in total. These K partial-public models are then further aggregated to generate a public model. (Abbreviation SE\E/E). Alternatively, the initial federation process may be performed directly on the local models of the multiple tenants at each edge to generate M partial-public models. A public model is then generated by combining all of the partial-public models in a second federation process. (Abbreviation SE/E\E).

When utilizing the edge-cloud for training, local models are first trained for each tenant in the edge. Federation may then be performed in various ways. For example, a partial-public model may be created directly in the cloud by aggregating the local models of each tenant from all the edge nodes. A second federation process is then performed to aggregate the partial-public models of each tenant to produce a public model. (Abbreviation SE\C/C). Alternatively, the first federation process may aggregate the local models of all the tenants at each edge node in the cloud to produce M partial-public models. A second federation process is then performed to aggregate all these partial-public models to produce a public model. (Abbreviation SE/C\C).

3) 3-Federation: In 3-federation, small-public services perform local training in the fog. First federation may be performed in either the fog or the edge, while the second and third federations may be performed in the edge or the cloud.

For the case where first federation is performed in the fog, the local models of each tenant are aggregated to produce MK partial-public models. Second and third federation may then be performed in two ways. For example, the partial-public models of each tenant at the different fog nodes may be aggregated to generate K partial-public models, and these K partial-public models are then further aggregated to create a public model in third federation. (Abbreviation SF\F\E/E) Alternatively, second federation may be performed to aggregate the local models of all the tenants at each edge to generate M partial-public models. Third federation is then performed to aggregate these M partial-public models to generate a public model. (Abbreviation SF\F/E\E).

First federation in the fog may also be performed among all the tenants at each fog node. In particular, the local models of all the tenants at each fog node are aggregated to produce a total of MN partial-public models. Second federation is then performed to aggregate the models of all the fog nodes under the same edge node. Finally, third federation is performed to aggregate the public models at all the different edge nodes to generate a public model. (Abbreviation SF/F\E\E).

For the case where the first and second federations are performed at the edge, the local models of each tenant are aggregated at each edge node to create MK partial-public models. Second and third federation can then be performed in two ways. For example, the partial-public models of each tenant at the different edge nodes can be aggregated to generate K partial-public models, and these partial-public models can then be further aggregated across all the tenants to generate a public model. (Abbreviation SF\E\E/E). Alternatively, second federation can be performed to aggregate the partial-public models of the various tenants at each edge node to produce M partial-public models, and these partial-public models can then be aggregated once again across the different edge nodes to produce a public model. (Abbreviation SF\E/E\E). As a third option, the SF\E/E\E arrangement described above can be reconfigured such that the first federation is performed in the edge rather than in the fog. (Abbreviation SF/E\E\E).

For the case where the first and second federations are performed in the edge and cloud, respectively, the local models of each tenant are first aggregated at the edge to generate MK partial-public models. Second and third federation may then be performed using the same SF\E\E/E and SF\E/E\E arrangements as those described above, with the exception that both federation processes are performed in the cloud rather than in the edge. (Abbreviations SF\E\C/C and SF\E/C\C). Finally, federation in the edge may also be performed across all the tenants. Second and third federation may then be performed using the same SF/E\E\E arrangement as that described above, with the exception that both federations are performed in the cloud rather than in the edge. (Abbreviation SF/E\C\C).

D. Detection

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

E. Mapping Examples

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

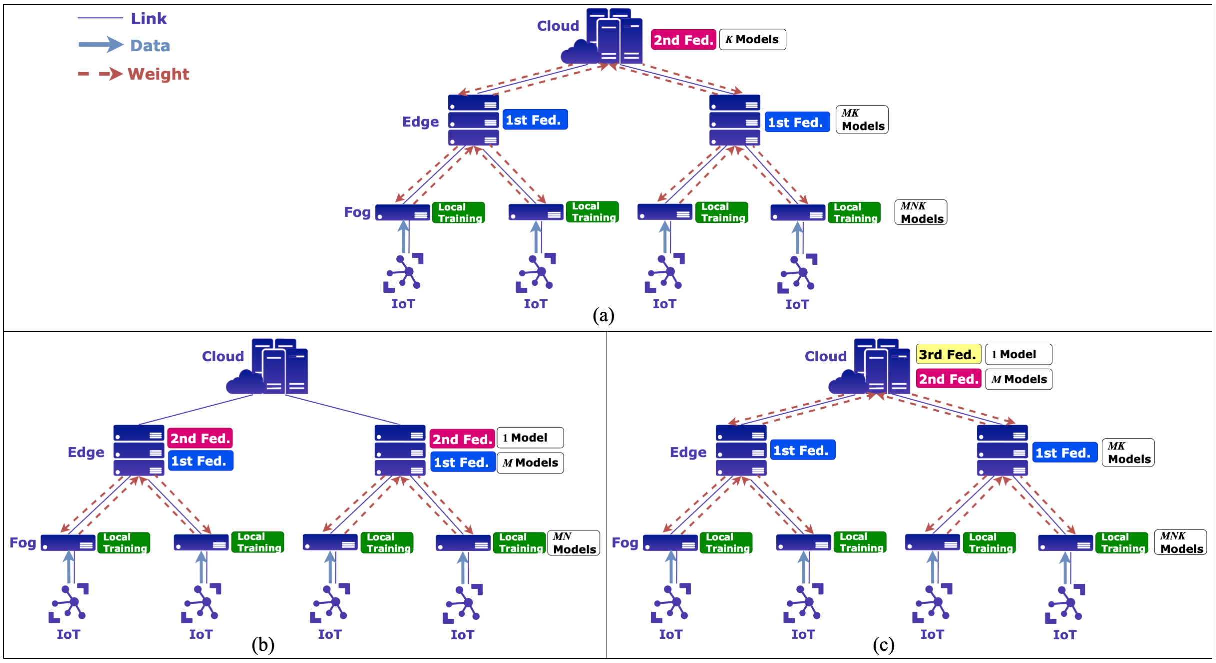

Fig. 2: Illustrative mapping examples: a) small-private SF\E\C; b) big-public BF\E\E; c) small-public SF\E/C\C.

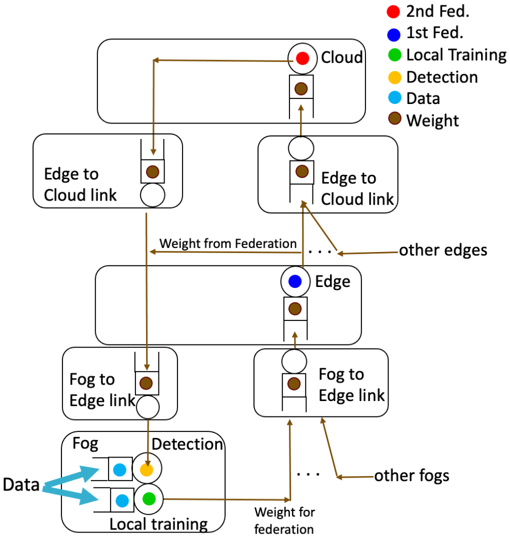

Fig. 3: Operational flow of illustrative BF\E\C system.

Figure 3 presents a detailed schematic of the BF\E\C system operation flow for illustration purposes with the detection process is 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 generates a weight data for saving the model characteristics. The weight of local models are then sent to the edge for first federation with the weight of the other fog nodes. After the first federation, the new weights are pushed back to all the fogs to update the respective local models. The weights are additionally forwarded to the cloud for the second federation to generate a public model. Finally, the public model is pushed back to all the fogs to update the local models and detection models.

VII. Open Challenges

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

A. Architecture Optimization

Architecture optimization is an essential activity aimed at minimizing the resources required while satisfy the AIoT latency constraints. As described above, a total of 31 possible job assignments have been identified. Among those assignments, the SF\E\E and BF\E\E configurations are the most promising candidates for small-private and big-public services since they employ hierarchical FL with 2-federation, which achieves a shorter model training time and lower energy cost than conventional FL [15]. Among all of the small-public task arrangements described above, the SF\E\C/C, SF\E/C\C, and SF/E\C\C services appear to be the most promising since they perform 3-federation which facilitate a rapid update of the local models. Furthermore, a three-tier structure allows the heavy task of FL to be offloaded to a higher tier, thereby minimizing the computation delay.

However, further optimization studies are still required to determine the optimal job assignment for each service which minimizes the delay and/or capacity.

B. Machine Learning

The performance of AIoT is crucially dependent on the ML model used to analyze the IoT data. Thus, the prediction performance of the ML model must be optimized through the selection of appropriate training methods and tuning

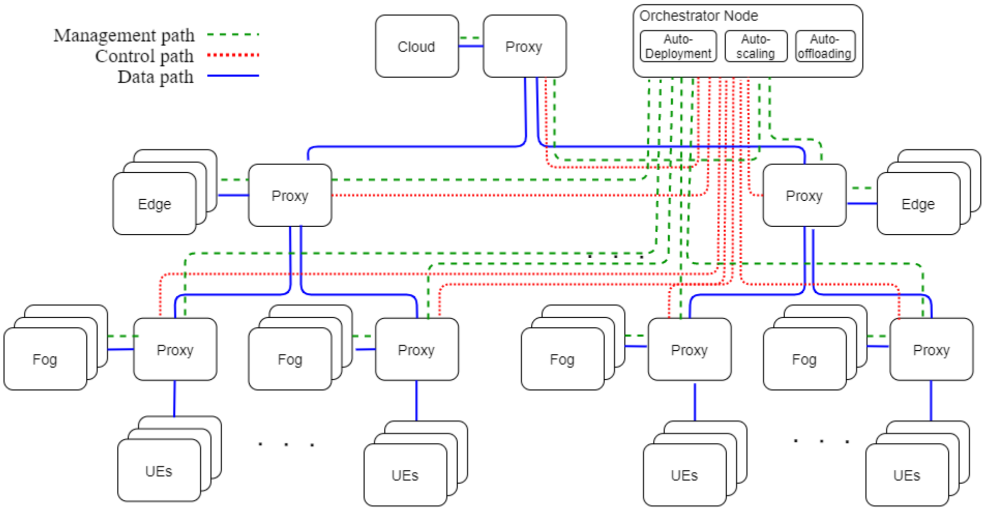

Fig. 4: Illustrative service framework architecture.

hyperparameters. In FL, one of the most important hyperparameters is the frequency of the training updates. In particular, excessively frequent updates can strain the computing resources, while infrequent updates may reduce the ML performance.

C. Security

When implementing AIoTaS, the network and architecture must be protected at all costs. 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.

D. Service Framework

To properly manage AIoT services running on distributed networks, some type of centralized service framework is required to resolve the scalability and availability issues which may arise. For example, an orchestrator may be used to manage and control the fog, edge, and cloud through various plugins, such as auto-deployment for automating the AIoT service deployment to the architecture. Moreover, auto-scaling and offloading may be required to automate the resource scaling and traffic offloading tasks in response to detect performance degradations, such as a longer processing time for user requests. In addition, a proxy may be required in each tier to assist the orchestrator in deploying services, and performing offloading. Accordingly, effective service frameworks, such as that shown in Figure 4 for illustration purposes, must be developed to realize the full commercial potential of AIoTaS.

VIII. Conclusion

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 services have been identified, comprising 7 small-private services, 7 big-public services, and 17 small-public services. Previous studies have suggested that small-public services have nearly the same accuracy as small-private services. Thus, it is anticipated that most AIoTaS applications might be deployed as small-public services, with only highly sensitive applications deployed as small-private services. 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 secure the data transmissions and ML model in the FL process and to develop an effective service framework for managing the operations of the distributed AIoTaS architecture.

References

[1] 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 *2020 IEEE 19th International Symposium on Network Computing and Applications (NCA)*, 2020, pp. 1–8.

[2] F. Firouzi, B. Farahani, and A. Marinšek, "The convergence and interplay of edge, fog, and cloud in the AI-driven Internet of Things (IoT)," *Information Systems*, vol. 107, p. 101 840, 2022.

[3] T. Liu, J. Xia, X. Wei, T. Wang, X. Fu, and M. Chen, "Efficient Federated Learning for AIoT Applications Using Knowledge Distillation," *arXiv*, vol. abs/2111.1, 2021. [Online]. Available: https://arxiv.org/abs/2111.14347.

[4] X. Zhang, M. Hu, J. Xia, T. Wei, M. Chen, and S. Hu, "Efficient federated learning for cloud-based aiot applications," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 11, pp. 2211–2223, 2021.

[5] W. Liu, H. Lin, X. Wang, *et al.*, "D2mif: A malicious model detection mechanism for federated learning empowered artificial intelligence of things," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[6] A. Li, L. Zhang, J. Wang, *et al.*, "Efficient federated-learning model debugging," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, 2021, pp. 372–383.

[7] C.-H. Chen and C.-T. Liu, "A 3.5-tier container-based edge computing architecture," *Computers & Electrical Engineering*, vol. 93, p. 107 227, 2021.

[8] 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 Network*, vol. 33, no. 2, pp. 58–64, 2019.

[9] E. Raj, D. Buffoni, M. Westerlund, and K. Ahola, "Edge MLOps: An Automation Framework for AIoT Applications," in *2021 IEEE International Conference on Cloud Engineering (IC2E)*, 2021, pp. 191–200.

[10] 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 Communications Surveys Tutorials*, vol. 22, no. 3, pp. 1722–1760, 2020.

[11] 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 of Things Journal*, vol. 8, no. 18, pp. 13 849–13 875, 2021.

[12] 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 Network*, vol. 35, no. 5, pp. 74–82, 2021.

[13] P. Kairouz, H. B. McMahan, B. Avent, *et al.*, *Advances and Open Problems in Federated Learning*. 2021.

[14] M.-T. Thai, Y.-D. Lin, Y.-C. Lai, and H.-T. Chien, "Workload and capacity optimization for cloud-edge computing systems with vertical and horizontal offloading," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 227–238, 2020.

[15] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6. DOI: 10.1109/ICC40277.2020.9148862.