Federated Threat-Hunting Approach for Microservice-Based Industrial Cyber-Physical System

Mohamed Abdel-Basset, *Senior Member, IEEE*, Hossam Hawash, and Karam Sallam, *Member, IEEE*

***Abstract*—The lightning convergence of industry 4.0 and the intelligent Internet of Things (IoT) technologies has significantly increased the vulnerability of industrial cyber-physical systems (ICPSs) to a large population of cyber threats. Intelligent threat detection for discovering cyber threats is a challenging task as it essentially deals with wide-scale, complicated, and heterogeneous ICPSs. This article presents a novel federated deep learning (DL) model (Fed-TH) for hunting cyber threats against ICPSs that captures the temporal and spatial representations of network data. Then, a container-based industrial edge computing framework is designed to deploy the Fed-TH as a threat-hunting microservice on suitable edge servers while maintaining decent resource orchestration. To tackle the latency issue of an ICSP, an exploratory microservice placement method is introduced to enable better microservice deployment based on the computational resources of the participants. The simulation results obtained from two public benchmarks validate the effectiveness of these approaches in terms of accuracy (92.97%, 92.84%) and f1-scores (91.61%, 90.49%).**

***Index Terms*—Cyber-physical system, deep learning (DL), industrial Internet of Things (IIoT), threat intelligence (TI).**

I. Introduction

AN INDUSTRIAL cyber-physical system (ICPS) is commonly known as a combination of physical procedures, networking, and computation in industrial environments. The rapid evolution of industry 4.0 has promoted flexible data communication through the Internet of things (IoT) networks and empowered the operability and productivity of ICPSs [1]. The construction of an ICPS involves different IoT technologies, such as software-defined networking, fifth-generation (5G) and beyond 5G (B5G) networks, cloud computing [2], mobile edge computing (MEC) [3], and standard industrial control systems [4]. The heterogeneous, wide-scale, distributed, and intricate nature of an ICPS causes many common and application-related vulnerabilities that can be abused by attackers to execute malicious actions [4]. While CPSs manage physical activities, the effect of any malicious action might be irrevocable, which depends greatly on the acuteness of the damage and the application's sensitivity [5].

Nowadays, 5G is considered a promising technology that will perform a critical role in delivering communications for autonomous industrial IoT (IIoT) applications [6]. It increases the reliance of an ICPS on cellular systems for delivering highly efficient services that satisfy the strict obligations of industry 4.0 in terms of short delays, ultraconsistency and scalability [3]. Also, the availability of beam-forming possibilities and broad bandwidths facilitates the interconnection of billions of heterogeneous IIoT devices in an ICPS. Therefore, the 5G technology promotes the provision of services for time-sensitive and task-crucial applications. However, this large-scale and heterogeneous connectivity imposes more security challenges for an ICPS [3], [5], [6].

Of many solutions to the current security challenges, deep learning (DL) has shown promising performances for developing efficient threat intelligence (TI) in IIoT networks; for example, a convolutional neural network (CNN) is known for its efficient feature extraction [7] and long short-term memory (LSTM) [8] and a gated recurrent unit (GRU) for temporal dependency modeling [9]. An autoencoder (AE) enables the efficient extraction of hidden representations from input while reducing dimensionality [10]. Most current studies assume that DL models are trained centrally with data generated by heterogeneous industrial terminals expanding exponentially [2], [3]. However, the transmission of such immense amounts of data to a prevailing cloud-based TI service makes industrial data more prone to cyber threats and induces a larger delay that is intolerable for evolving ICPSs [6].

As a proliferating paradigm, MEC intends to bring computations from the cloud to a network's edges to minimize the network's latency and bandwidth congestion while preserving the privacy of data owners. Motivated by MEC's promising advantages of dexterity, instantaneous processing and independence, efforts are increasinglyfocusing on the development of

Manuscript received February 18, 2021; revised April 27, 2021 and June 6, 2021; accepted June 17, 2021. Date of publication June 22, 2021; date of current version December 6, 2021. Paper no. TII-21-0731. *(Corresponding author: Hossam Hawash.)*

The authors are with the Faculty of Computers and Informatics, Zagazig University, Zagazig 44519, Egypt (e-mail: mohamed.abdelbasset@fci.zu.edu.eg; hossamreda@zu.edu.eg; karam_sallam@zu.edu.eg).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TII.2021.3091150.

Digital Object Identifier 10.1109/TII.2021.3091150

edge-cloud ICPSs [8], [9]. To improve an ICPS's total performance, heterogeneous edge devices offload rigorous communication services from the distant cloud and place the processing burden on a resource-restricted device, i.e., onsite industrial equipment. Despite the great advances in resource distribution achieved by DL, it is challenging to develop a distributed DL on empowered edge nodes owing to resource constraints and privacy matters, including data eavesdropping and/or leakage. In this regard, federated learning (FL) is an encouraging approach that has achieved great success in delivering intelligence to the edge layers of IoT networks with participating devices using local data to autonomously train a global model predownloaded from a central authority. Sharing local updates instead of raw data improves the communication proficiency of FL models while reducing privacy issues.

International IIoT marketplaces have been witnessing a significant tendency to be more service-enabled and/or have a distributed design, which entails breaking down these complicated systems into reasonably distinctive and integral containerized microservices. These microservices could constitute a complete system that might be stateless or have an exclusive data-consistency tier [8]. A containerization design offers potential advantages concerning implementation and deployment, responsiveness, scalability, and efficiency, as validated through its significant approval by various pioneering industrial technology corporations, such as Amazon, Tesla, and Google [11]. This makes a microservice design a candidate solution for the design of a decentralized attack/intrusion detection system for ICPSs. Nevertheless, the additional complexity intrinsic in a microservice-based design also introduces numerous issues concerning its efficiency and effectiveness. In this regard, this article emphasizes the development of a microservice-based TI solution for the reliable and efficient detection of cyber attacks across containerized edge nodes.

A. Main Challenges

Several challenges facing the development of efficient microservice-based intelligent threat hunting for an ICSP are briefly described as follows.

1) **Security:** The convergence of an ICPS and 5G network allows malicious agents to use the well-known susceptibilities of a cellular network to launch cyber threats which potentially have a negative effect on the industrial process [3].
2) **Heterogeneity:** A 5G-enabled ICPS usually comprises a wide variety of attributes in various industrial things, such as the cloud, edge servers (ESs), applications and data, which greatly hinder the capabilities of a TI model [8], [11]. Also, a microservice design is prone to large slowdowns and/or run-time collisions because of the heterogeneity of ESs stemming from high variations in transient network interrupts, memory footprints, and the frequencies of central processing unit (CPU) cycles. However, the standard allocation, deployment and supervision of containerized microservices did not consider the heterogeneity of either physical or virtualized nodes. Moreover, the recuperation capacity of Kubernetes is mostly achieved by observing the containers' states, nodes and pods, and reviving/overcoming defeats, which is insufficient for high accessibility.
3) **Privacy:** Maintaining the privacy of industrial data is a crucial concern when developing any DL-based application [12]. Current FL cannot fully guarantee the preservation of the privacy of participating entities as the central authority (cloud) is often presumed to be completely trustworthy for managing training. Nevertheless, the central authority can simply breach the privacy of local contributors by monitoring the relevant behaviors and upgrades using recent cyber attacks [13].
4) **Latency:** As an ICPS often entails time-critical tasks, rapid responses and low latency values are necessary [14].

B. Primary Contributions

This study's contributions to solving the abovementioned challenges by proposing a new threat-hunting approach for IIoT networks are briefly discussed in the following.

1) A DL model (Deep-TH) for detecting cyber threats in an ICPS is presented. Multiscale convolutions are introduced to capture spatial representations and a GRU-based AE to obtain temporal ones.
2) For the first time, a container-based industrial edge computing (CIEC) framework is introduced to incorporate the Deep-TH in a differentially private FL framework referred to as Fed-TH.
3) This framework promotes integrating the Fed-TH under a microservice-based edge deployment using a new exploratory placement method that improves latency.

C. Article Organization

The rest of this article are organized as follows. In Section II, related research studies. In Section III, an architectural design of our system. In Section IV, the proposed Fed-TH framework. In Section V, the methodology of our experiments. In Section VI, results, comparisons, analyses, and discussions. After that, Section VII argue the main advantages and disadvantages of the proposed solution. Finally, Section VIII concludes this article

II. Literature Review

A. Cyber-Threat Intelligence for ICPSs

Lately, increasing research attention has been paid to the intelligent detection of cyber threats/attacks against ICPSs. In this context, Li *et al.* [9] introduced a new DL approach for recognizing intrusions in ICPSs ,which employs a GRU and CNN to effectively learn the representations of different classes of attacks from sequential IoT data. They also presented an FL schema for enabling data to be exploited from geographically distributed CPS vendors. Hussain *et al.* [1] developed a combined framework that integrates residual CNN and real network data to quickly discover distributed denial of service (DDoS) threats using a botnet for malevolent device management. Zhou *et al.* [7] introduced a few-shots learning approach based on

the Siamese CNN to relieve the over-fitting problem and improve the performance of smart intrusion detection in an ICPS. Farivar *et al.* [4] employed a neural network (NN) to act as an intelligent online method for cyber threat/attack estimation and reconstruction in legacy nonlinear ICSs. Zhou *et al.* [10] addressed intrusion detection using a variational LSTM model that employs an encoder-decoder architecture accompanied by a variational method to learn the low-dimensional patterns from high-dimensional IoT data. However, these studies still had the following three main shortcomings.

1) They ignored the data's heterogeneity during training which is common for ICPSs, especially those running on 5G networks.

2) The majority of ICPS-related TI methods were designed based on a strong hypothesis that enough excellent samples of cyber threats on ICPSs were constantly obtainable for designing intelligent detection models. However, realistically, one ICPS terminal typically has few cyber-threat samples, which makes the development of a DL model incredibly challenging.

3) ICPS holders are customarily unwilling to disclose their data (normal or malicious) to third parties due to the sensitivity and privacy of the information they often contain. Therefore, developing a reliable DL model for cyber-threat hunting or detection in ICPSs is a complex challenge.

B. Microservice-Based IoT Applications

Microservice-based solutions have been developed to improve the flexibility of IoT networks by shifting application designs to another level using innovative standards of accountability, scalability, swift evolvement, and flexible implementation [10]. As an alternative to a single monolithic design that incorporates all application modules as a single entity, the microservice design breaks them into numerous microservices for self-evolving services in the context of development and functional supervision [11]. Given this, Coulson *et al.* [15] presented a pipeline for the automatic scaling of microservice-based applications by experimenting with a supervised DL model for advocating appropriate scaling activities. Similarly, Guangba *et al.* [16] developed a scaling system for finding the optimal scaling requirements for services to satisfy the relevant service-level agreement through an acceptable computational cost. Zhao *et al.* [17] presented a redundant placement strategy, which employs a sample average approximation to transmit microservices into heterogeneous ESs, with the main aim to realize faster response times. Wang *et al.* [14] developed a placement strategy for an edge-cloud architecture to address the latency of smart manufacturing applications. Chen *et al.* [11] presented a deep reinforcement learning (RL) technique for optimizing the deployment of microservices in dynamic and heterogeneous cloud-edge systems. Samanta *et al.* [18] introduced a mathematical method for scheduling microservices in EC systems to reduce the costs and delays in IoT networks. Abdullah *et al.* [19] implemented a new DL approach for the automatic allocation of CPU resources to containers (i.e., dockers and Kubernetes) to realize the optimal number of parallel tasks and achieve an efficient performance.

Unfortunately, none of these studies considered a microservice design for TI applications. Also, the development of smart industrial services consisting of multiple microservices with intrinsically complicated dependencies still require careful planning to achieve the efficient latency of TI services on the edge side of a network.

III. System Design

The design of the framework of the proposed system consists of three primary components, i.e., containerized edge nodes (e.g., ESs or mobile devices), a cloud server, and IoT network, as shown in Fig. 1.

A. Cloud Backend

The cloud backend is responsible for constructing a final threat-hunting model by federating the parameters of an edge-trained one. Numerous cycles of communication among the cloud server and industrial participants (IPs) to collaboratively obtain a complete and efficient threat-hunting model are conducted. The cloud backend generally manages the distributed training of Fed-TH, as discussed later, and consists of the following four main distinct modules.

1) Resource Manager (RM): The main role of the RM is to retain the resource profile of each Industrial agent involved in federated training. The system takes into account edge resources in terms of computation, memory, storage, input/output (I/O) communication, and processing.

2) Service Manager (SM): The primary responsibility of the SM is to retain the profile of every active service/microservice. The system emphasizes microservice-level data involving multihop workflows (i.e., microservice reliance) and resource obligations.

3) Scheduler (Sc): The Sc is liable for deciding on the strategy for deploying threat-hunting microservices on specific industrial devices to reduce the overall delay in the system while ensuring that the overall restrictions are satisfied at the same time as the microservices [14]. This implies that placing more microservices on edge nodes is feasible to realize more real-time threat hunting and microservices are placed on the cloud only when there are deficient edge resources [17].

4) Deployer (Dp): The Dp is accountable for deploying and initiating all threat-hunting microservices based on the placement strategy established using the Sc.

B. Containerized Edge Tier

ESs are computational entities of CIEC responsible for delivering remote edge-based microservices for hunting cyber threats in an ICSP. More explicitly, after obtaining the microservice scheduling strategy from the cloud, the Dp deploys the container images like a code fountain, runs the Fed-TH and, finally, sends the outcomes back to the IPs. The ESs communicate with the IPs through a direct channel to decrease broadcast dormancy. Moreover, the CIEC is scalable and flexible as the ESs can be easily attached or detached by the SM depending on physical requirements or health circumstances.

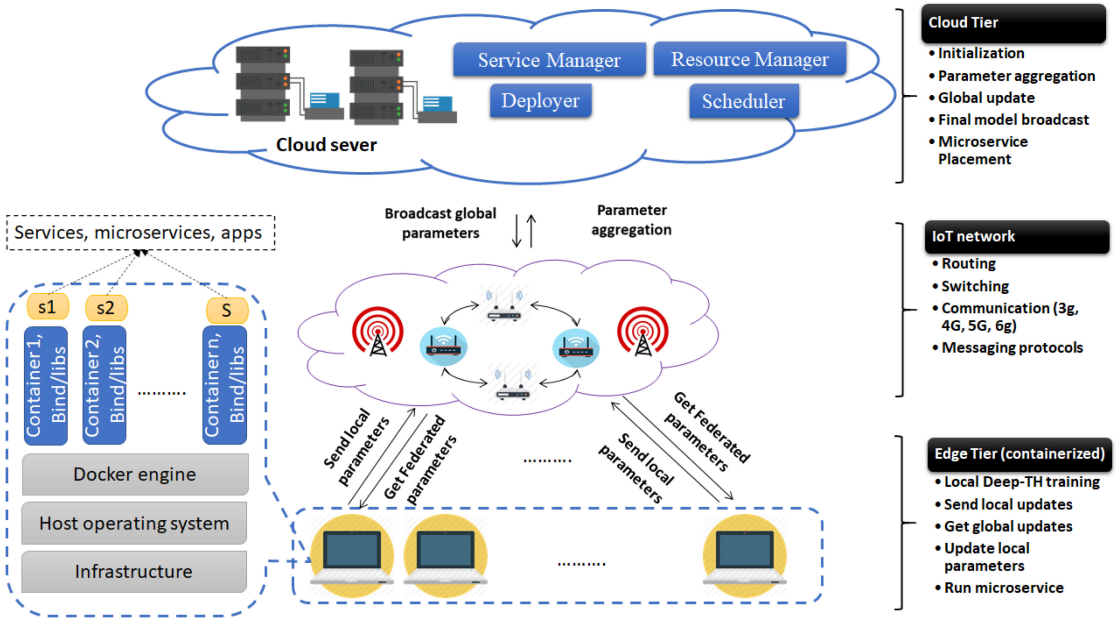

Fig. 1. Systematic diagram indicating the system model of proposed for federated threat/attack detection in microservice-based ICPS, where the threat hunting models are deployed locally at containerized edge server (Edge tier),and the global model simuletenously deployed at the cloud server (Cloud tier). Both Edge tier, Industrial agents, and Cloud Tier are communicating via IoT network established using different combination of comunicationtion technologies and messaging protocols. The cloud tier take on the responsibility of managing microservices at edge server using diversity of modules.

C. Microservice Placement

For the efficient offloading of a threat-hunting microservice in the CIEC framework, an exploratory microservice placement (EMP) method (Algorithm 1) is introduced to compare the computational delays and local and remote discharging, thereby competitively choosing the offloading policy for optimal performance regarding delays.

Typically, the microservice offloading policy entails two procedures. 1) An offloading policy determination whereby, for each mission created with a particular industrial entity, the local offloading wait is contrasted with the remote offloading one, respectively and the offloading policy with the shortest latency determined. 2) A microservice placement whereby, for the remote offloading strategy, the algorithm computes the EMP method and edge offloading wait. Finally, the remote computational dormancy is transmitted for the offloading policy judgment.

Local latency ($L(t)$) is considered when the threat-hunting task is performed locally with no communication overhead. It depends on the required number of CPU cycles ($\varphi(t)$) for task $t \in T = \{t^1, t^2, \cdots, t^n\}$ and the processing capacity of the IPs. Therefore, when the free capability ($\mathrm{Fr(IP)}$) is larger than the necessary memory ($\mu(t)$), the $L(t)$ is computed by

$$\mathrm{LL}(t) = \begin{cases} \varphi(t)/P(\mathrm{IP}) & \text{if } \mathrm{Fr(IP)} > \mu(t) \\ \infty, & \text{otherwise} \end{cases}. \tag{1}$$

Once the tasks are offloaded to the CIEC system, the microservices are assigned priority scores (PSs) based on their processing attributes and data reliance. Later, they are queued according to these scores and condensed in containers ($C(t) = \{c^1, c^2, \cdots, c^k\}$). Then, the remote latency comprises: 1) processing time (T^{Proc}) which represents the containers' running times in ESs; and 2) the communication time (T^{Comm}), which is the time taken to transmit the data from the IPs to the ESs through a 5G network. The PS of is calculated by

$$\mathrm{PS}(m) = T^E_{\min}(ms) + \max_{\mathrm{ms}^s \in \mathrm{post(ms)}} (\mathrm{ms}^s) \tag{2}$$

where $T^E_{\min}(m)$ represents the minimum execution time, that is, $T^E_{\min}(m) = \max_{E^i \in E} \frac{\varphi(\mathrm{m})}{P(E^i)}$. Naturally, microservices that are nearer to the entry and require fewer CPU cycles are assigned higher scores. Then, the threat-hunting microservices can be ranked and condensed in decreasing order. Given a container's completion/end time (T^{end}_C), the start time $[T^{\mathrm{start}}(c, \mathrm{ES})]$ of container c on the ES can be calculated by

$$T^{\mathrm{start}}(c, E) = \max\left(A(E), \max\left(T^{\mathrm{end}}(c) + \omega(c, c^p)\right)\right) \tag{3}$$

where $A(E)$ denotes the free time of an ES to run a container and $\omega(\cdot, \cdot)$ the communication delay between c and c^p, with $c^p \in \mathrm{pre}(C)$. Then, the allocation status of each container is determined by

$$f(c, E) = \begin{cases} 1 & \text{if } c \text{ is placed on } E \\ \infty, & \text{otherwise} \end{cases}. \tag{4}$$

The ES's completion/end time ($T^{\mathrm{end}}(E^i)$) for the containers stacked in a queue (Q^i) is equal to T^{end}_C of the last queued

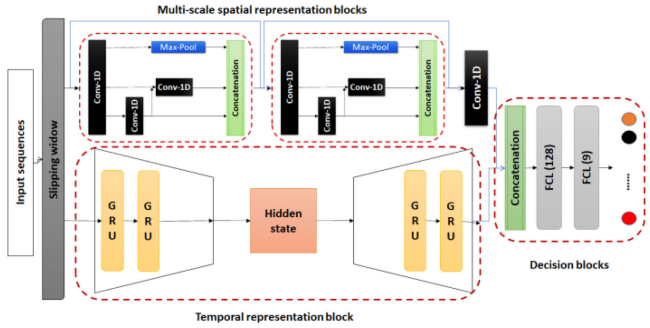

Fig. 2. Architecture of proposed Deep-TH model (MSR blocks (upper part) extract spatial representations and TL block temporary ones, with both representations concatenated and final class computed by decision block (right-hand part)).

container, as formulated by

$$T^{\text{end}}\ \left(E^i\right) = \max_{c \in C} \left\{ \left(T^{\text{start}}\left(c, E^i\right) + \frac{\varphi\left(c\right)}{P\left(E^i\right)} \right) f\left(c, E^i\right) \right\}. \tag{5}$$

Therefore, the total processing time (T^{Proc}) of threat hunting is computed as the longest completion time of all the ESs by

$$T^{\text{Proc}} = \max_{\text{ES}^i \in \text{ES}} T^{\text{end}}\ \left(E^i\right)\ . \tag{6}$$

Conversely, the communication time (T^{Comm}) for task t is computed by

$$T^{\text{Comm}} = \frac{\propto (t)}{\text{SH}\left(\text{IP}, E\right)} \tag{7}$$

where $\propto (t)$ represents the data size of t and $\text{SH}(\text{IP}, E)$ the Shannon-Hartley transmission rate. Then, the remote latency ($\text{RL}(t)$) is calculated by

$$\text{RL}\left(t\right) = T^{\text{Proc}} + T^{\text{Comm}}. \tag{8}$$

Then, the calculated local and remote latencies are compared to determine the best placement.

IV. Proposed Fed-TH

In this section, the design of the proposed model for cyber threat hunting in an edge-based ICPS is described. In Fig. 2, a systematic diagram of its structural design, which is comprised of three main blocks: 1) multiscale spatial representation (MSR); 2) temporal learning (TL); and 3) decision blocks. Where the detailed describtio of these blocks is presented in the following subsections.

A. Multiscale Spatial Representation Block

The main role of this block is to learn efficient spatial representations in ICPS data sequences. To accomplish this, the time series are prepared with a slipping window to transform univariable input into a 2-D representation. Then, a hierarchical stack of residually connected multiscale 1-D convolutions for efficient spatial feature extraction, where the rectified linear unit is employed as the activation function, is presented. As the high-level spatial representations obtained might miss or leak information across layers, an inner residual link using the max-pooling layer to enable feature usage while focusing more on the most important features is added. Given that the output of the spatial slipping window is S^w, that of the first Conv-1D is calculated by

$$F = \text{Conv} - \text{ID}_k\left(S^w\right) \tag{9}$$

where k is the kernel size. Then, the output of the MSR block is calculated by

$$O = \text{Concat}\left(\hat{F}_i\right), \quad \text{for } i = 1, 2, 3 \tag{10}$$

where $\hat{F}_i$ is calculated as the output of three paths according to

$$\hat{F}_i = \begin{cases} \text{MaxPool}\left(f_i\right) & i = 1 \\ \text{Conv} - 1D\left(F_i\right) & i = 2 \\ \text{Conv} - 1D\left(F_i + \hat{F}_{i-1}\right), & i = 3 \end{cases} \tag{11}$$

where F_i is a feature map of path i and $\hat{F}$ an output map computed from each path. In the proposed model, two MSR blocks are stacked to learn spatial representations in the data.

Algorithm 1: EMP Method.

Input: a set of tasks ($T = \{t^1, t^2, \cdots, t^l\}$) , a set of ESs ($E = \{E^1, E^2, \cdots, E^m\}$) and a set of IPs ($\text{IP} = \{\text{IP}^1, \text{IP}^2, \cdots, \text{IP}^n\}$)

For all t in T **do:**
  If $\text{Fr}(\text{IP}) > \mu(t)$ **then:**
    Calculate the local latency LL(t) by (1)
  Else
    $L(t) = \infty$
  Assign PSs for of the microservices
  Condense them into containers ($C(t) = \{c^1, c^2, \cdots, c^k\}$)
  For $i = 1$ *to* k **do:**
    Create array ($T^{\text{end}}[\,]$);
    For $j = 1$ *to* m **do:**
      If $\mu(c^i) \geq Fr(E^j)$ **then:**
        Set $T^{\text{end}}[\,j] = \infty$.
      Else
        Calculate the $T^{\text{end}}[\,j]$ by (5).
      If $i = 1$ **then:**
        Calculate the T^{Comm} between IP and ES^j.
        $T^{\text{end}}[\,j] = T^{\text{end}}[\,j] + T^{\text{Comm}}$.
    Assign the c^i to the ES with the smallest end time.
    $T^{\text{end}}\ (c^i) = \text{minimum}\ (\text{T}^{\text{end}}\ )$
    Upgrade the free resources of the ESs and IPs.
  $\text{RL}(t) = \max_{\text{c} \in \text{C(t)}} T^{\text{end}}\ (c)$
  If $LL(t) \leq RL(t)$ **then:**
    Execute t locally
  Else
    Execute t on the CIEC servers.
  Upgrade the related resources for the next task.

B. Temporal Learning (TL) Block

This is a different enhanced TL block that, for the first time, involves an AE and GRU (AE-GRU) inspired by a Recurrent Neural Network-based AE. Given the input time series as $x = \{x^1, x^2, \ldots, x^t\} \in \mathbb{R}^{n\times t}$, where n represents the input length and t the time step, the encoder takes the input and processes it by stacking the GRU layers. Then, the encoder's hidden state is upgraded by

$$h^t = g\left(h^{t-1}, x^t\right) \tag{12}$$

where $h^t \in \mathbb{R}^s$ represents the encoder's hidden state, s is the hidden dimension, and g its nonlinear activation. The fundamental notion behind a GRU is its capability to capture temporal dependency while maintaining memory efficiency. It employs an update gate (z^t) to regulate the amount of new data to be elapsed in the present state and a reset gate (r^t) to regulate the amount of data available from the intranet state. A candidate activation can be regarded as the new information at the current time, with the r^t used to control the amount of historic data to be maintained. It is generated by the z^t and candidate activation, with the former controlling how much new and old information is retained. The encoding process is formulated as follows:

$$z^t = \sigma\left(W^z x^t + W^z h^{t-1} + b^z\right) \tag{13}$$

$$r^t = \sigma\left(W^r x^t + W^r h^{t-1} + b^r\right) \tag{14}$$

$$\hat{h}^t = \tanh\left(W^{\hat{h}} x^t + r^t \circ W^{\hat{h}} h^{t-1} + b^{\hat{h}}\right) \tag{15}$$

$$h_E^t = \left(1 - z^t\right) \circ h^{t-1} + z^t \circ \hat{h}^t \tag{16}$$

where σ, $\tanh$ and $\circ$ represent the sigmoid activation, tanh activation and Hadamard product, respectively, W^z , W^r, and $W^{\hat{h}}$ the weight matrices, b^z, b^r , and $b^{\hat{h}}$ the gates' bias vector and h_E^t the $t-th$ hidden encoding state. Finally, the hidden state is formulated as

$$V = \left[h_E^t, h_E^2, \cdots, h_E^m\right] \in \mathbb{R}^{m\times d}. \tag{17}$$

The hidden state can be deemed a form of halfway information that can be used for decoding operations. Therefore, the decoder typically acts to extract temporal patterns in the intermediate representations for an improved classification performance. Although a fully connected layer (FCL) is often employed to achieve this, GRUs are deployed for decoding and extracting high-level representations. The hidden information with double the dimensions of the encoder is fed to the decoder. The generated decoder's hidden state (h_D^t) includes temporal representations of the input sequences. Then, the outputs of the TR and MSR modules are concatenated r and passed to subsequent blocks for a decision regarding the final cyber-threat class (Fig. 2).

C. Decision Block

This block takes the extracted representations and processes them to obtain the final classification decision. This is accomplished using two FCLs and SoftMax to generate the probability score that the input belongs to a given $i$thclass as

$$P^i = e^i / \sum_j^{\mathrm{C}} e^j \tag{18}$$

where C represents the number of output classes. The categorical cross-entropy (CCE) loss, which is used to train the proposed model, is calculated by

$$\mathrm{Loss} = \mathrm{CCE} = -\frac{1}{M}\sum_i^M Y^i \log\left(P^i\right) \tag{19}$$

where Y^i and P^i represent the actual label and estimated probability for the $i-\mathrm{th}$ class, respectively, and M the total number of classes.

D. Federated Training

Each ES trains its own Deep-TH locally on its local dataset using the adaptive stochastic gradient descent algorithm. It is assumed that each ES retains the recent local gradients (lg_i) for submission to the cloud for each communication iteration as

$$lg_i{}^t = \nabla L_i\left(w_i{}^t\right) \tag{20}$$

where ∇L_i represents the loss function on the $i-th$ ES with parameter w_i. The local parameter of th Deep-TH in the $t-th$ iteration is computed by

$$w_i^t = w_i^{t-1} + \propto^t \left(lg_i{}^{t-1}\right) . \tag{21}$$

The cloud server retains the most recent global gradient (g^{global}) to be broadcast back to the ESs as

$$gg^{\,t} = \frac{1}{NB}\sum_{i=1}^N lg_i^t \tag{22}$$

where t represents the global iteration, B the batch size and N the number of ESs. The cloud sets the initial g^{global} to zero and then handles the gradient download and uploads transactions from the ESs. Once an upload request (g_i^{local}) is received, it is automatically added to the g^{global}. When the cloud server receives a download request from any ES, it first filters the g^{global} and then delivers it to that ES. Therefore, the local Deep-TH models indirectly interact together through the cloud server, which is responsible mainly for averaging the learned gradients from various ESs and delivering the result back to them.

As this forward-backward transmission incurs a large communication cost, we propose selecting the valuable gradients (i.e., filtered ones) during the uploading and downloading processes by

$$g_i^t = \left\{\vartheta^g \in \overrightarrow{\left|g_{i,j}^t\right|}, \vartheta^g \in \overrightarrow{V\left(g_{i,j}^t\right)}\right\} \tag{23}$$

where $|g_{i,j}^t|$ represents the value of the $j-\mathrm{th}$ parameter and $V(g_{i,j}^t)$ the related variance between two successive iterations sorted in incremental order. This can be demonstrated for calculations of both global and local gradients.

Unlike previous studies [9], [20], [21] that employed computationally complex encryption methods to protect the privacy of local parameters, the local gradients of the Deep-TH are

Algorithm 2: Federated Training of Fed-TH.

Input: N subsets of database $\{D^1, D^2, \cdots, D^N\}$, batch size B.

Output: Federated optimal parameters (**w**) of Fed-TH.

1: Set initial parameters w^0 for all participants.
2: $t = 0$.
3: **ESs perform.**
4: **For** $i = 1$ *to* N **do:**
5: The ith ES request gradients (gg^{t-1}) from cloud.
6: The local parameters (w_i^t) updated by (21).
7: Calculate the noised parameter $(\overline{\overline{w}}_i^t)$ using (24) to achieve LDP.
8: The $i$thES compute the local gradients $lg_i{}^t$ by (20).
9: Upload the filtered gradient by (23).
10: $t = t + 1$
11: **Cloud performs.**
12: Aggregate gradients form ESs.
13: Calculate the mean gradient (gg^{t}) by (22).
14: Broadcast filtered gradient (gg^t) by (23) to the ESs.
15: Repeat from lines 3 to 14 until the Fed-TH converges.
16: Return the final parameters $(\boldsymbol{w})$ of Fed-TH.

safeguarded by the local differential privacy (LDP) [22] method before being submitted to the cloud. To safeguard the privacy of the upgraded parameters of the Deep-TH and realize LDP, a Gaussian mechanism [23], [21] is applied to the local Deep-TH of each ES via combining noise to realize the parameters' perturbation as

$$\overline{\overline{w}}_i^t = w_i^{t-1} + \propto^t \left(lg_i{}^{t-1} + \mathcal{N}\left(0\ ,\ s_h^2\sigma^2\right)\right) \tag{24}$$

where $\propto^t$represents the step size toward the negative gradient and $\mathcal{N}(0\ ,\ s_h^2\sigma^2)$ the incorporated noise with an average of zero and standard deviation of $s_h.\sigma$. Therefore, the incorporation of LDP forces the ES to locally train the Deep-TH with a noisy parameter $(\overline{\overline{w}}_i^t)$ through the $t-\text{th}$ iteration and, to guarantee $\epsilon-\text{privacy}$, the overall privacy charge should be constrained in ϵ. Given the total number of iterations as T, the privacy charge needs to be divided evenly throughout the iterations so that $\epsilon = \sum_1^T \epsilon^t$ but,when that total charge in the $t-\text{th}$ iteration exceeds ϵ, i.e., $\sum_1^t > \epsilon$, the training process must be terminated.

Considering the improvements in accuracy and privacy charges across iterations, the privacy charge in every iteration is adaptively assigned, with the initial privacy cost for training set to ϵ^0. When the g_i^{local} donates a great deal into the g_t^{global}, extra noise must be incorporated to safeguard confidentiality as the gradients are converging. Therefore, the privacy charge at the $t-th$ iteration is $\epsilon^t = \epsilon^0.(1-\propto)$ and, conversely, when the g_i^{local} donates little, a higher privacy charge is applied as $\epsilon^t = \epsilon^0.(1-\beta)$, which implies that a lower level of noise is applied to the gradients. The pseudo-code of the training procedure is presented in Algorithm 2.

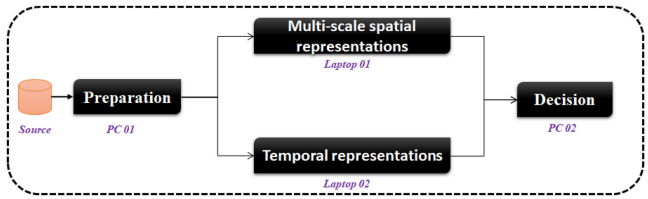

Fig. 3. DAG representations of Fed-TH service in which placements of all waiting microservices should be decided according to EPM strategy.

E. Modularization

After the training procedures are completed, the proposed Fed-TH is considered ready to use a TI service, which is then analyzed and modularized into numerous microservices according to the directed acyclic graph (DAG) representations presented in Fig. 3. The microservices can be deployed on any edge device based on the availability of both resource and location requirements. Remarkably, a lightweight transportation protocol, i.e., the message queuing telemetry transport (MQTT), is employed to effect transmissions among different microservices. Then, the proposed EPM method is used to offer efficient placements of microservices across industrial edge devices.

V. Experimentation Strategy

In this section, the methodology for conducting the experiments in this study is clearly discuss under three different aspects including the empirical settings, evaluation datasets, and performance measures.

A. Empirical Settings

The simulation experiments are conducted on a Dell personal computer equipped with a graphical processing unit (NVIDIA Quadro K2200), a CPU (Intel (R) Xeon (R) CPU E5-2670 0@ 2.60GHz) with 256 GB of memory and a 64-bit Windows 10 operating system. The implementation process for the models is coded in a Python 3.7 environment using a PyTorch[1] library. The Kubernetes[2] is employed for the management and orchestration of Docker[3] containers. Also, the optimal hyperparameters of the proposed Fed-TH are determined based on grid-search experiments and reported in Table I.

B. Descriptions of Datasets

Different from current homogeneous datasets, which do not reflect the real-world behaviors of IoT data, the proposed model is evaluated using the ToN_IoT [8] and LITNET-2020 datasets [24]. The former contains aggregated labeled IoT/IIoT data including heterogeneous data sources belonging to nine classes

[1][Online]. Available: https://pytorch.org/
[2][Online]. Available: https://kubernetes.io/
[3][Online]. Available: https://www.docker.com/

TABLE I
HYPERPARAMETERS OF PROPOSED FED-TH

Hyperparameter	Value
Batch size	128
No. of epochs	150
No.r of rounds	20
Optimizer	Adam
Initial learning rate	0.007
No. of GRU layers	2
No. of MSR blocks	2

of IoT traffic, that is, normal, scanning, DoS, ransomware, backdoor, injection, and cross-site scripting (XSS) attacks as well as a password cracking attack (PWA) acquired using seven IIoT devices. Its distribution information is presented in Table II and more detailed descriptions of experiments using it and its features are available in [8]. The LITNET-2020 dataset comprises 85 network flow features corresponding to 12 attack types, where five network exporters are employed to monitor and acquire network flows. In Table III, its main characteristics and distributions are shown, with more details of its acquisitions and features provided in [24].

C. Data Prepreparation

To guarantee efficient training of the model, some preprocessing steps are applied to the raw data. First, the values of the categorical variables are encoded into numeric values using a label-encoding technique[4] . Also, to avoid the effect of large discrepancies between the values of different features, min–max normalization is applied to rescale them into the range of [01] as

$$x_{\text{normalized}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{25}$$

where x, $x_{\max}$, and$x_{\min}$ represent the value of the current instance, the minimum value of x and the maximum value of x, respectively. To mitigate the impact of class imbalance, the adaptive synthetic sampling technique [25] is employed to adaptively generate more synthetic samples for the minority of classes based on their distributions. Simultaneously, the majority of classes are downsampled to their lowest numbers of samples. The data are divided into two portions of 80% for training and 20% for testing purposes. Moreover, for FL experiments, the training data are equally distributed across the containerized edge nodes.

D. Performance Measures

To measure the performance of the proposed model, the accuracy, precision, recall, and the F1-score are defined by

$$\text{Accuracy } (A) = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \times 100 \tag{26}$$

$$\text{Precision } (P) = (\text{TP}/(\text{TP} + \text{FP})) \times 100 \tag{27}$$

$$\text{Recall } (R) = (\text{TP}/(\text{TP} + \text{FN})) \times 100 \tag{28}$$

[4][Online]. Available: https://scikit-learn.org/stable/index.html

$$F1 - \text{score } (F1) = 2{*}P{*}R/P + R \tag{29}$$

where FN, FP, TP, and TN represent false negative, false positive, true positive and true negative samples, respectively.

VI. EXPERIMENTS AND ANALYSIS

A. Results

The experiments conducted in this study evaluate the proposed framework under a multiclass scenario, which is more challenging than the binary classification of IoT data (either normal or attack). The confusion matrices for the ToN_IoT and LITNET-2020 datasets are shown in Tables IV and V, respectivel, where the precision of each class is presented in the last row and its recall and F1-score in the last two columns, respectively. In Table IV, it can be observed that the normal and XSS classes have high precision values of 94.88% and 92.10%, respectively, while those of the others are similar, between 89.98% and 91.64%. In Table V, it is clear that the code red worm and TCP SYN-flood classes achieve high precision values of 93.96% and 93.59%, respectively, and the others those of between 87.91% and 90.82%. These results demonstrate the capabilities of the proposed Fed-TH to efficiently recognize different forms of cyber-physical attacks.

B. Comparative Analyses

To demonstrate the efficiency of the proposed Fed-TH framework, comparisons of it with cutting-edge federated DL methods for threat detection are performed [9], [26]. These approaches are reimplemented based on their settings reported in corresponding papers and evaluated under the same experimental settings as the proposed framework to guarantee fair comparisons. For the same purpose, their centralized architectures are reimplemented under the same federated training scheme of our model. The results obtained from evaluating these competing methods on the test sets of the previously mentioned datasets are presented in Table VI. Using the ToN_IoT dataset, it is observed that the proposed Fed-TH achieves a significant performance improvement (accuracy 2.6% and F1-score 1.6%). Similarly, it outperforms the other methods using the LITNET-2020 dataset, with improvements of 2.2% and 1.9% for accuracy and the F1-score, respectively. This explains the capability of the Fed-TH to model both the spatial and temporal knowledge intrinsic in ICPS data.

A paired *t*-test is performed to assess the statistical importance of the results based on a significance threshold value, that is, $p - \text{value} = 0.05$, where $p - \text{values} < 0.05$ indicate that the Fed-TH's results are statistically different from those of the competing models, with Table VII displaying their computed *p*-values. It can be seen that most *p*-values are less than the threshold, which validates the statistical significance of the results obtained by the Fed-TH.

C. Ablation Studies

In Fig. 4, the results of ablation experiments performed to analyze the impacts of different building blocks, where the

TABLE II
CHARACTERISTICS OF TON_IOT DATASET

Device	No. of samples	Features	Class	Distribution
Fridge	587076	Ts,Date,Time,Fridge_tempreature, Temp_condition,Date,Time, Door_state,Sphone_signal, Latitude,Longitude, Motion status, Signal status, FC1_Read_Input_Register,FC2_Read_Descret_ Value, FC3_Read_Holding_Register, FC4_Read_Coil,Current_temperature, Thermostat_status, Temperature, Humidity, Pressure, Label, Type	Backdoor	246136
Garage	591446		Ddos	53992
GPS	595686		Injection	50319
Modbus	287194		Normal	3086973
Motion	452261		Password	142674
Thermostat	442228		Ransomware	16030
Weather	650242		Scanning	3973
/	/		Xss	6037

TABLE III
CHARACTERISTICS OF LITNET-2020 DATASET

Exporter	Features	Class	Distribution
Kaunas–Vytautas Magnus University and Kaunas Technological University (KTU), Vilnius Gediminas Technical University, Klaipeda University, Siauliai University, KTU Panevezys Faculty of Technologies and Business Kaunas University of Technology	ts_year, ts_month, ts_day, ts_hour, ts_min, ts_second, te_year, te_montht,_day, te_hour, te_min, te_second, td, sa, da, sp, dp, pr, _flag1, __flag2, _flag3,_flag4, _flag5,_flag6, fwd, stos, ipkt, ibyt, opkt, obyt, _in, out, sasdas, smk,dmk, dtos, dir, nh, nhb, svln, dvln, ismc, odmc, idmc, osmc, mpls1, mpls2, mpls3, mpls4, mpls5, mpls6, mpls7, mpls8, mpls9, mpls10, cl, sl, al, ra, eng, exid, tr, icmp_dst_ip_b,icmp_src_ip udp_dst_p, tcp_f_s, tcp_f_n_a, tcp_f_n_f, tcp_f_n_r, tcp_f_n_p, tcp_f_n_u, tcp_dst_p, tcp_src_dst_f_s, tcp_src_tftp, tcp_src_kerb, tcp_src_rpc,tcp_dst_p_src, smtp_dst, dp_p_r_range, p_range_dst, udp_src_p.	Packet fragmentation attack (A1)	477
		Spam bot's detection (A2)	747
		Reaper Worm(A3)	1176
		Scanning/Spread(A4)	6232
		ICMP-flood(A5)	11,628
		HTTP-flood (A6)	22,959
		Blaster Worm(A7)	24,291
		LAND attack(A8)	52,417
		Smurf(A9)	59,479
		UDP-flood(A10)	93,583
		Code Red Worm(A11)	1,255,702
		TCP SYN-flood (A12)	3,725,838

TABLE IV
CONFUSION MATRIX OF FED-TH ON TON_IOT DATASET

		Predicted Classes									
		Backdoor	DDos	Injection	Normal	Password	Ransomware	Scanning	XSS	**R** (%)	**F1** (%)
Actual Classes	Backdoor	44476	301	101	3433	289	131	287	209	90.35%	90.91%
	DDos	211	27905	305	1647	167	227	199	137	90.61%	90.29%
	Injection	105	222	27898	1123	188	234	193	101	92.80%	91.95%
	Normal	2871	1891	1245	206427	1134	1453	1131	1243	94.95%	94.92%
	Password	305	103	271	1156	26094	159	249	198	91.45%	91.27%
	Ransomware	183	129	323	1417	354	26163	313	324	89.58%	90.03%
	Scanning	243	209	206	1139	205	341	28209	243	91.60%	91.62%
	XSS	228	254	266	1213	211	207	201	28627	91.73%	91.92%
	P (%)	91.47%	89.98%	91.13%	94.88%	91.10%	90.48%	91.64%	92.10%		

TABLE V
CONFUSION MATRIX OF FED-TH ON F LITNET-2020 DATASET

		Predicted Classes													
		A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	**R**(%)	**F1**(%)
Actual Classes	A1	10095	15	8	24	5	7	46	11	16	2	573	493	89.38	89.58
	A2	5	10276	6	3	11	7	0	6	18	2	502	513	90.55	90.10
	A3	7	22	12940	39	18	31	29	103	44	16	502	484	90.90	90.48
	A4	71	89	62	14483	33	47	28	27	31	51	511	613	90.26	90.54
	A5	24	59	18	15	14968	11	9	6	7	8	599	602	91.68	90.76
	A6	113	84	23	4	8	13409	3	14	7	5	488	434	91.89	91.12
	A7	89	21	21	2	5	15	15282	11	3	0	613	796	90.65	90.49
	A8	34	54	88	11	19	0	57	14608	33	55	822	702	88.62	89.84
	A9	43	97	31	24	0	10	23	5	15884	7	819	953	88.76	88.33
	A10	16	29	14	13	18	7	8	16	28	17612	1109	1847	85.01	86.94
	A11	323	302	415	717	774	516	813	615	913	1039	236442	8271	94.15	94.06
	A12	424	413	743	612	798	779	621	614	1084	999	8648	229433	93.58	93.59
P (%)		89.78	89.66	90.05	90.82	89.86	90.36	90.32	91.10	87.91	88.97	93.96	93.59		

TABLE VI
COMPARATIVE ANALYSIS OF DIFFERENT FEDERATED DL MODELS USING DIFFERENT EVALUATION METRICES

	ToN_IoT Dataset				LITNET-2020 Dataset			
Model	A	P	R	F1	A	P	R	F1
LSTM [8]	88.43%	88.13%	87.29%	87.71%	87.04%	86.01%	86.06%	86.03%
CNN [1]	86.13%	85.97%	87.03%	86.50%	85.37%	85.64%	84.77%	85.20%
AMCNN-LSTM [26]	89.67%	89.88%	90.02%	89.95%	87.97%	88.13%	87.31%	87.72%
Deep-Fed [9]	90.32%	90.06%	90.11%	90.09%	90.68%	88.83%	88.45%	88.64%
Fed-TH	**92.97%**	**91.60%**	**91.63%**	**91.61%**	**92.84%**	**90.53%**	**90.45%**	**90.49%**

TABLE VII
P-VALUES FROM PAIRED *T*-TEST EXPERIMENT

Model	ToN_IoT Data		LITNET-2020 Data	
	A	F1	A	F1
LSTM [8]	1.347E-02	2.013E-02	2.144E-02	9.133E-03
CNN [1]	2.135E-02	2.414E-02	3.127E-02	1.432E-02
AMCNN-LSTM [26]	9.982E-03	1.935E-02	2.135E-02	1.432E-02
Deep-Fed [9]	4.123E-02	2.384E-02	3.412E-02	2.131E-01

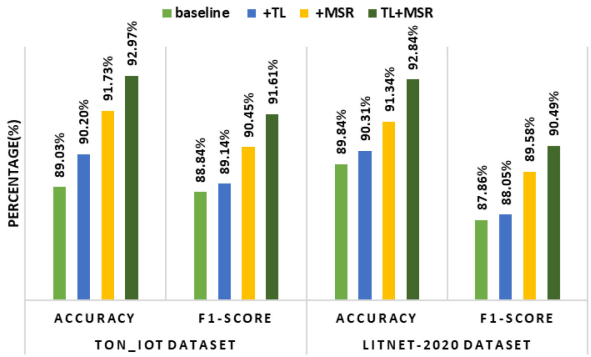

Fig. 4. Results of ablation experiments obtained from proposed Fed-TH.

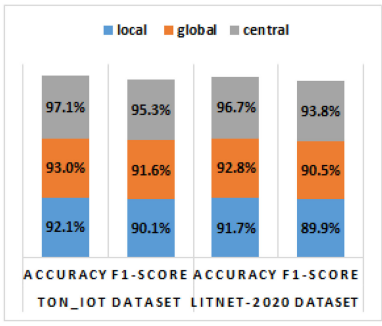

Fig. 5. Classification performances under federated, local and central Fed-TH.

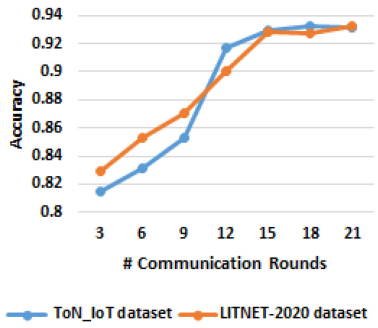

Fig. 6. Accuracy under different numbers of communication rounds.

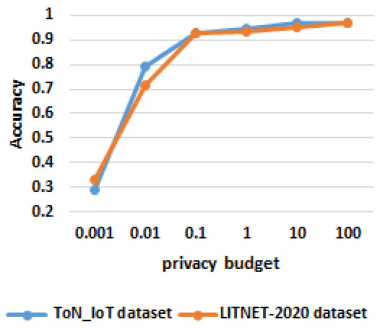

Fig. 7. Classification accuracy under different privacy budgets.

LSTM-AE [27] is employed as a baseline model, are presented. For both datasets, it can be noted that the TL block improves performances with their accuracy and F1-score values both 1%. Also, the MSR block shows significant improvements (accuracy 2%, F1-score 2%) over the baseline. This explains the effectiveness of multiscale residual convolutions for improving classification performances while integrating both blocks obtains powerful performance improvements (accuracy: 3%, F1-score 3%) over the baseline. This indicates that both temporal and spatial features are essential and complementary for robust cyber-threat detection.

D. Federated Versus Central Learning

Apart from the previous experiments, the performance of the proposed Fed-TH is compared with those of local variants built locally using a little data and with the centrally built version of the Fed-TH trained on all data samples. In Fig. 5, the arithmetical results of these comparisons in terms of accuracy and F1-score values are displayed. Notably, the local variants exhibit the worst performances while the federated model achieves a satisfactorily good performance relative to that of the central one. This indicates that the proposed Fed-TH provides a threat-intelligence solution suitable for all ICPS owners because of its high efficiency, awareness of heterogeneity and privacy-preserving characteristics.

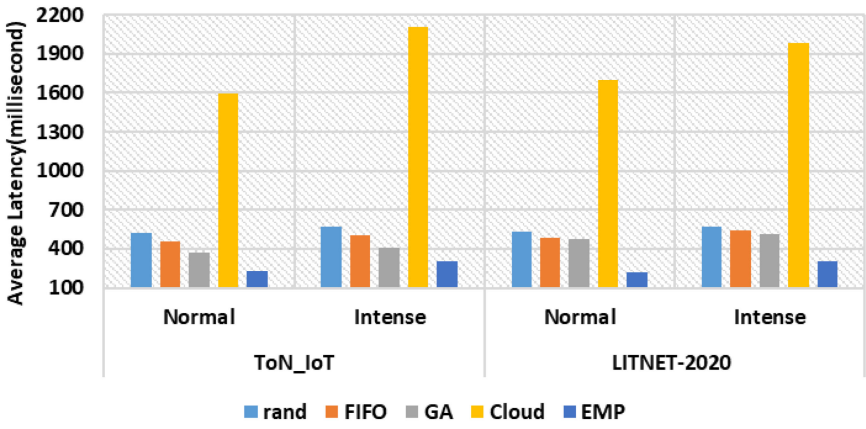

Fig. 8. Comparison of average latency performances of different placement methods.

E. Number of Communication Rounds

An additional experiment is performed to analyze the number of communication rounds required for convergence during the training procedures. The accuracy of the proposed Fed-TH under a different number of rounds using the abovementioned datasets while the other settings are fixed is shown in Fig. 6. Notably, the Fed-TH converges after 15 and 12 rounds of communication on the ToN_IoT and LITNET-2020 data, respectively. This rapid convergence further implies that it has an efficient communication overhead.

F. Tradeoff Between Privacy and Accuracy

The proposed Fed-TH is evaluated to analyze its variations in accuracy under various privacy budgets. Typically, obtaining greater privacy necessitates a smaller value of ϵ which implies that more Gaussian noise must be added. As seen in Fig. 7, the relationship between the model's privacy and accuracy obviously requires a tradeoff. It is notable that the privacy budget $\epsilon = 0.1$ results in a satisfactory classification, therefore alleviating such a tradeoff, which is a critical opportunity for investigation in future work.

G. Microservice Placement in CIEC Framework

In this experiment, the performances of microservice placements are evaluated in terms of average latency. It is assumed that the arrival rate of each IP follows the Poisson procedure using factor λ [28], where normal traffic flows are generated with $\lambda = 500$ and intense ones with $\lambda = 2000$.

1) Comparative Analysis: The average latency attained by the proposed EMP is compared with those of the placement methods random, first-in-first-out (FIFO) [29] and genetic algorithms [11], as shown in Fig. 8. It is also compared with cloud placements, where all containers are allocated to cloud servers. It can be seen that a cloud placement obtains the worst latency owing to its communication delay. The random and FIFO methods achieve high latency values in the CIEC framework because, as they are unaware of network information and resource consumption, they result in unproductive placements.

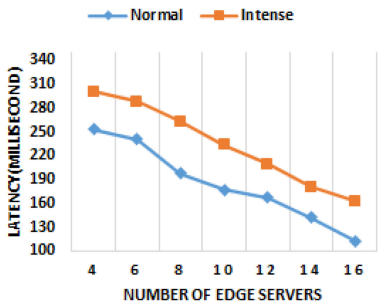

Fig. 9. Average latency values with different numbers of ESs under normal and intense flows obtained from ToN_IoT data.

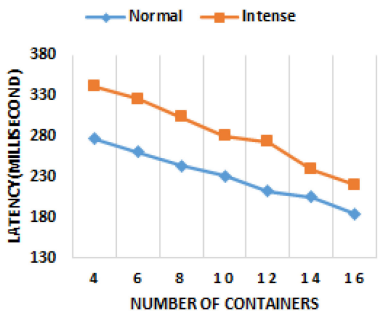

Fig. 10. Average latency values with different numbers of containers under normal and intense flows obtained from ToN_IoTdata.

2) Impact of Number of ESs: In Figs. 9 and 12, the associations between the average latency and number of ESs for the ToN_IoT and LITNET-2020 data, respectively, are displayed. It is notable that increasing the number of ESs decreases the average latency until it stabilizes. This indicates that the average latency of the EMP method depends on a limited number of ESs and becomes steady when there are sufficient ESs.

3) Impact of Number of IPs: In Figs. 10 and 13, the relationships between the number of IPs and average latency for the ToN_IoT and LITNET-2020 data, respectively are shown. It is notable that the average latency of the EMP method remains stable with increases in the number of IPs. This is because the it

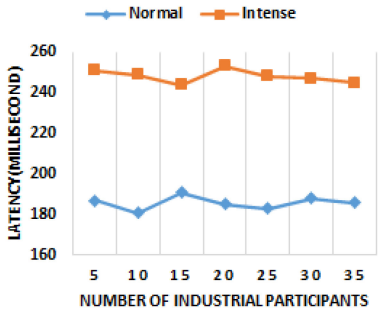

Fig. 11. Average latency values with different numbers of IPs under normal and intense flows obtained from ToN_IoTdata.

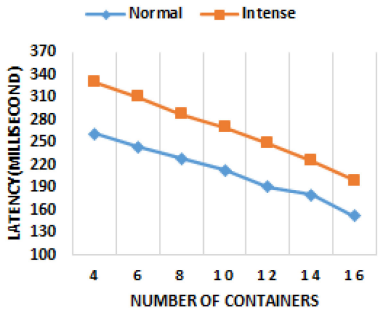

Fig. 13. Average latency values with different numbers of containers under normal and intense flows obtained from LITNET-2020 data.

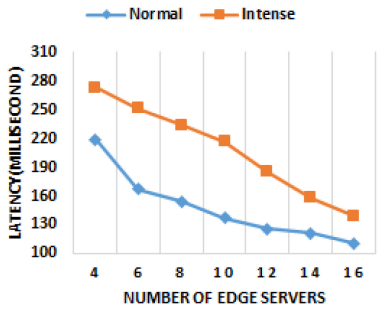

Fig. 12. Average latency values with different numbers of ESs under normal and intense flows obtained from LITNET-2020 data.

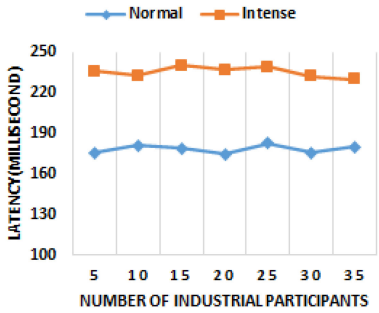

Fig. 14. Average latency values with different numbers of IPs under normal and intense flows obtained from LITNET-2020 data.

accumulates all the information and calculates the appropriate deployment policies, thereby not incurring additional latency even with an intense traffic flow.

4) Impact of Number of Containers: In Figs. 11 and 14, the relationships between the average latency and number of containers for the ToN_IoT and LITNET-2020 data, respectively, are presented. With increases in the number of containers, the average latency improves as the EMP method evaluates the resource consumption of the ESs and calculates the most suitable container placements. The remote latency is shorter than the local one as the computational resources of ESs are much more powerful than those of IPs and, therefore, the EMP method can realize improved latency.

VII. Advantages and Disadvantages

The main advantages of this work are three-fold. First, it considers the design of a federated threat-detection model for complex and heterogeneous data. Second, it is the first attempt to consider deploying a federated model as a threat-hunting microservice using latency-aware placements which improves the real-time performance of threat detection in an ICPS. Third, it is the first study to introduce a differentially private FL model for detecting cyber attacks in containerized heterogeneous edge computing devices. On the other hand, the main limitations of this work are as follows. First, widely available unlabeled data is still unable to be exploited during federated training which might limit the amount of heterogeneity in the data. Second, differential privacy might impose some negative impacts on detection performances owing to the noise applied to the data. Third, the proposed system presumes that all the participants are trusted and suitable to engage in the training, whereas malicious or improper ones can cause catastrophic consequences for an ICPS.

VIII. Conclusion

In conclusion, this article presented a novel federated DL approach (i.e., the Fed-TH) for efficient cyber-threat detection in a ICPS. Collaborative training in the edge-cloud environment revealed its efficiency while preserving the privacy of participants. A CIEC framework was introduced to integrate the Fed-TH in a microservice-based architecture to provide latency-effective placements using the EMP method. Extensive analyses revealed the efficiency of this framework in terms of accuracy and latency. For future improvements, load balancing in multicloud IIoT environments will be explored. Also, a severance deployment strategy for installing a single TI microservice on various ESs might be an expansion of this article. The tradeoff between productivity and privacy might be investigated using a blockchain and the client selection dilemma is likely to be studied in the context of federated training.

References

[1] B. Hussain, Q. Du, B. Sun, and Z. Han, "Deep learning-based DDoS-Attack detection for cyber-physical system over 5G network," *IEEE Trans. Ind. Informat.*, vol. 17, no. 2, pp. 860–870, Feb. 2021.

[2] S. Wang, Z. Ding, and C. Jiang, "Elastic scheduling for microservice applications in clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 1, pp. 98–115, Jan. 2021.

[3] F. Spinelli and V. Mancuso, "Towards enabled industrial verticals in 5G: A survey on MEC-based approaches to provisioning and flexibility," *IEEE Commun. Surv. Tut.*, vol. 23, no. 1, pp. 596–630, Nov. 2020.
[4] F. Farivar, M. S. Haghighi, A. Jolfaei, and M. Alazab, "Artificial intelligence for detection, estimation, and compensation of malicious attacks in nonlinear cyber-physical systems and industrial ioT," *IEEE Trans. Ind. Informat.*, vol. 16, no. 4, pp. 2716–2725, Apr. 2020
[5] N. Slamnik-Krieštorac, H. Kremo, M. Ruffini, and J. M. Marquez-Barja, "Sharing distributed and heterogeneous resources toward end-to-end 5G networks: A comprehensive survey and a taxonomy," *IEEE Commun. Surv. Tut.*, vol. 22, no. 3, pp. 1592–1628, Jun. 2020.
[6] T. Jiang *et al.*, "3GPP Standardized 5G channel model for IIoT scenarios: A survey," *IEEE Internet Things J.,* vol. 8, no. 11, pp. 8799–8815, Jun. 2021.
[7] X. Zhou, W. Liang, S. Shimizu, J. Ma, and Q. Jin, "Siamese neural network based few-shot learning for anomaly detection in industrial cyber-physical systems," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5790–5798, Aug. 2021.
[8] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, and A. Anwar, "TON_IoT telemetry dataset: A new generation dataset of IoT and IIoT for data-driven intrusion detection systems," *IEEE Access*, vol. 8, pp. 165130–165150, Sep. 2020.
[9] B. Li, Y. Wu, J. Song, R. Lu, T. Li, and L. Zhao, "DeepFed: Federated deep learning for intrusion detection in industrial cyber-physical systems," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5615–5624, Aug. 2021.
[10] X. Zhou, Y. Hu, W. Liang, J. Ma, and Q. Jin, "Variational LSTM enhanced anomaly detection for industrial big data," *IEEE Trans. Ind. Informat.*, vol. 17, no. 5, pp. 3469–3477, May 2021.
[11] L. Chen *et al.*, "IoT microservice deployment in edge-cloud hybrid environment using reinforcement learning," *IEEE Internet Things J.,* to be published, doi: 10.1109/JIOT.2020.3014970.
[12] S. Rathore and J. H. Park, "A blockchain-based deep learning approach for cyber security in next generation industrial cyber-physical systems," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5522–5532, Aug. 2021.
[13] J. Zhang, B. Chen, X. Cheng, H. T. T. Binh, and S. Yu, "PoisonGAN: Generative poisoning attacks against federated learning in edge computing systems," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3310–3322, Mar. 2021.
[14] Y. Wang *et al.*, "MPCSM: Microservice placement for edge-cloud collaborative smart manufacturing," *IEEE Trans. Ind. Informat.*, vol. 17, no. 9, pp. 5898–5908, Sep. 2021.
[15] N. C. Coulson, S. Sotiriadis, and N. Bessis, "Adaptive microservice scaling for elastic applications," *IEEE Internet Things J.,*vol. 7, no. 5, pp. 4195–4202, May 2020.
[16] G. Yu, P. Chen, and Z. Zheng, "Microscaler: Cost-effective scaling for microservice applications in the cloud with an online learning approach," *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/tcc.2020.2985352.
[17] H. Zhao, S. Deng, Z. Liu, J. Yin, and S. Dustdar, "Distributed redundancy scheduling for microservice-based applications at the edge," vol. 1374, pp. 1–14, 2019.
[18] A. Samanta and J. Tang, "Dyme: Dynamic microservice scheduling in edge computing enabled IoT," *IEEE Internet Things J.,*vol. 7, no. 7, pp. 6164–6174, Jul. 2020
[19] M. Abdullah, W. Iqbal, F. Bukhari, and A. Erradi, "Diminishing returns and deep learning for adaptive CPU resource allocation of containers," *IEEE Trans. Netw. Serv. Manage.*, vol. 17, no. 4, pp. 2052–2063, Dec. 2020.
[20] Y. Liu, J. J. Q. Yu, J. Kang, D. Niyato, and S. Zhang, "Privacy-preserving traffic flow prediction: A federated learning approach," *IEEE Internet Things J.,* vol. 7, no. 8, pp. 7751–7763, Aug. 2020.
[21] C. Zhou, A. Fu, S. Yu, W. Yang, H. Wang, and Y. Zhang, "Privacy-preserving federated learning in fog computing," *IEEE Internet Things J.,* vol. 7, no. 11, pp. 10782–10793, Nov. 2020.
[22] N. Rodríguez-Barroso *et al.*, "Federated learning and differential privacy: Software tools analysis, the Sherpa.ai FL framework and methodological guidelines for preserving data privacy," *Inf. Fusion*, vol. 64, pp. 270–292, 2020.
[23] M. Abadi *et al.*, "Deep learning with differential privacy," 2016, doi: 10.1145/2976749.2978318.
[24] R. Damasevicius *et al.*, "Litnet-2020: An annotated real-world network flow dataset for network intrusion detection," *Electron.*, vol. 9, 2020. [Online]. Available: https://doi.org/10.3390/electronics9050800
[25] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," 2008.
[26] Y. Liu *et al.*, "Deep anomaly detection for time-series data in industrial IoT: A communication-efficient on-device federated learning approach," *IEEE Internet Things J.,* vol. 8, no. 8, pp. 6348–6358, Apr. 2021.
[27] C. Yin, S. Zhang, J. Wang, and N. N. Xiong, "Anomaly detection based on convolutional recurrent autoencoder for IoT time series," *IEEE Trans. Syst. Man, Cybern. Syst.*, to be published, doi: 10.1109/tsmc.2020.2968516.
[28] L. Cui *et al.*, "A blockchain-based containerized edge computing platform for the internet of vehicles," *IEEE Internet Things J.,* vol. 8, no. 4, pp. 2395–2408, Feb. 2021.
[29] Y. Peng, Y. Bao, Y. Chen, C. Wu, and C. Guo, "Optimus: An efficient dynamic resource scheduler for deep learning clusters," in *Proc. 13th EuroSys Conf.,* 2018, pp. 1–14.