Received 10 October 2023; revised 21 November 2023; accepted 5 December 2023. Date of publication 12 December 2023; date of current version 5 January 2024.

*Digital Object Identifier 10.1109/OJCOMS.2023.3342008*

# A Vertical Heterogeneous Network (VHetNet)-Enabled Asynchronous Federated Learning-Based Anomaly Detection Framework for Ubiquitous IoT

**WEILI WANG[1,2,3], OMID ABBASI[2] (Senior Member, IEEE), HALIM YANIKOMEROGLU[2] (Fellow, IEEE), CHENGCHAO LIANG[3,4], LUN TANG[3,4], AND QIANBIN CHEN[3,4] (Senior Member, IEEE)**

[1]School of Cyber Security and Information Law, Chongqing University of Posts and Telecommunications, Chongqing 400065, China
[2]Non-Terrestrial Networks Lab, Department of Systems and Computer Engineering, Carleton University, Ottawa, ON K1S 5B6, Canada
[3]Chongqing Key Laboratory of Mobile Communications Technology, Chongqing University of Posts and Telecommunications, Chongqing 400065, China
[4]School of Communications and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China

CORRESPONDING AUTHOR: Q. CHEN (e-mail: cqb@cqupt.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant 62071078 and Grant 62001076.

**ABSTRACT** Anomaly detection for the Internet of Things (IoT) is a major intelligent service required by many fields, including intrusion detection, state monitoring, device-activity analysis, and security supervision. However, the heterogeneous distribution of data and resource-constrained end nodes in ubiquitous IoT systems present challenges for existing anomaly detection models. Due to the advantages of flexible deployment and multi-dimensional resources, high altitude platform stations (HAPSs) and unmanned aerial vehicles (UAVs), which are important components of vertical heterogeneous networks (VHetNets), have significant potential for sensing, computing, storage, and communication applications in ubiquitous IoT systems. In this paper, we propose a novel VHetNet-enabled asynchronous federated learning (AFL) framework by adopting the compound-action actor-critic (CA2C) algorithm for UAV selection, which enables decentralized UAVs to collaboratively train a global anomaly detection model based on their local sensory data from IoT devices. In the VHetNet-enabled AFL framework, the UAV selection process aims to prevent UAVs with low local model quality and large energy consumption from affecting the learning efficiency and model accuracy. Due to the wide coverage as well as strong storage and computation capabilities, a HAPS operates as a central aerial server for aggregating local models of UAVs asynchronously and making decisions intelligently. Moreover, we propose a CA2C-based joint device association, UAV selection, and UAV placement algorithm to further enhance the overall federated execution efficiency and detection model accuracy under UAV energy constraints. Extensive experimental evaluation on real-world datasets demonstrates that the proposed algorithm can achieve high detection accuracy with short federated execution time and low energy consumption.

**INDEX TERMS** Anomaly detection, asynchronous federated learning, compound-action actor-critic, ubiquitous Internet of Things (IoT), vertical heterogeneous network (VHetNet).

## ABBREVIATIONS

| | |
|---|---|
| AFL | Asynchronous Federated Learning |
| AI | Artificial Intelligence |
| CA2C | Compound-Action Actor-Critic |
| DDPG | Deep Deterministic Policy Gradient |
| DNN | Deep Neural Network |
| DQN | Deep Q-Network |
| DP | Differential Privacy |
| DRL | Deep Reinforcement Learning |
| FL | Federated Learning |

| | |
|---|---|
| HAPS | High Altitude Platform Station |
| IoT | Internet of Things |
| MDP | Markov Decision Process |
| ML | Machine Learning |
| RL | Reinforcement Learning |
| UAV | Unmanned Aerial Vehicle |
| VHetNet | Vertical Heterogeneous Network |
| WGAN-GP | Wasserstein Generative Adversarial Network with Gradient Penalty |

## I. INTRODUCTION

COMPARED to existing wireless technologies including 5G, 6G and beyond networks represent more than an improvement of basic performance requirements, such as increased data rates, reduced latency, and enhanced spectrum and energy efficiency [1]. 6G and beyond networks are envisioned to have some unique characteristics. First, space networks (including satellites in low, medium and geostationary Earth orbits), air networks (including high altitude platform stations (HAPSs) and unmanned aerial vehicles (UAVs)), and terrestrial networks (including macro and micro base stations), are integrated into vertical heterogeneous networks (VHetNets) to provide global coverage [2], [3]. Second, numerous devices, such as environmental monitoring sensors, healthcare wearables, and industrial control agents, are joining Internet of Things (IoT) networks. The ubiquitous IoT is expected to support seamless connectivity anytime, anywhere, and for everything [4]. Third, artificial intelligence (AI) penetrates into every corner of networks, ranging from end devices to the core. Network nodes are being endowed with built-in AI, which will support diversified AI services as well as facilitate intelligent network management [5].

Anomaly detection is defined as a process of automatically detecting whether devices, components, or systems are running normally or not [6]. Anomaly detection is one of the indispensable AI services required by IoT devices for early warnings of abnormal behaviors and uninterrupted operations. Data is the basis of anomaly detection, and machine learning (ML) is the most commonly used technique [7]. Traditional anomaly detection schemes in IoT usually utilize static sensors to perform data sensing and IoT gateways to forward the sensory data to a central server to learn a global ML model [8], [9], [10]. However, in situations where sensing targets are continually moving or are located in far-flung regions, traditional sensing solutions will experience significant challenges. In addition, due to the explosion of data in ubiquitous IoT, privacy concern and limited communication resources for data transmission present a major impediment to any centralized learning framework.

To implement anomaly detection in ubiquitous IoT, HAPSs and UAVs represent advanced approaches for smart sensing and data analysis [11], [12], [13], [14], [15]. A three-layer VHetNet consisting of a HAPS, UAVs, and IoT devices is a promising architecture for learning a global anomaly detection model, where UAVs are deployed as flying sensors for data sensing from IoT devices, and the HAPS is deployed as a central aerial server for data analysis and network control. Compared to static sensors, using UAVs as aerial nodes to provide wireless sensing support is a promising paradigm due to their wider field of view, highly flexible and controllable 3D mobility, and the capability of sensing performance optimization through UAV position adjustment [16]. In addition, a HAPS can work as an efficient central aerial server due to its quasi-stationary position, line-of-sight communication, wide coverage, and multiple energy sources, including conventional energy (electrical batteries and fuel tanks), energy beams, and solar energy [17].

As an emerging decentralized learning framework, federated learning (FL) has been shown to be effective in overcoming the challenges of privacy concerns and limited communication resources [4]. By using FL, a UAV can perform local model training based on sensory data from its associated IoT devices, and all local models are uploaded to the HAPS periodically for global aggregation. Compared to a centralized training framework, FL allows the VHetNet to learn a global anomaly detection model in a secure and efficient way. However, since there must be a periodic exchange of all UAV models with the HAPS during the model training process, the VHetNet-enabled FL framework still faces some challenges.

- The FL convergence will inevitably be affected by the learning latency and model quality of each UAV.
- Because UAVs and HAPSs operate in the sky and communicate through wireless communication technologies, their sensitive information is more likely to be inferred by the shared parameters between them.
- UAVs are generally energy-constrained, and hence balancing energy usage between tracking moving targets, computation, and transmitting data for model training is a thorny problem.

The challenges mentioned above require an in-depth investigation into the efficient implementation of anomaly detection in the VHetNet-enabled FL framework.

### A. RELATED WORK

In recent years, research on FL-based distributed learning frameworks has received much attention in wireless networks [18], [19], [20], [21], [22], [23], where the main role of FL has been to preserve data privacy and improve learning efficiency. Liu et al. [18], for instance, formulated an FL reinforcement learning framework in radio access network slicing to achieve an efficient device association scheme, where FL was adopted to promote the collaboration between smart devices while reducing the bandwidth consumption for learning. In [19], FL was used with edge intelligence–powered networks to decrease the signaling overhead and computing complexity of base stations when collaborating to learn a computation offloading

and interference coordination scheme. Wang et al. [20] proposed an FL-based edge caching framework to enable all local users to cooperatively learn a content popularity prediction model, which could accelerate the convergence speed and improve the hit rate. However, in [18], [19], [20], all local agents were required to participate in each global aggregation, which neglected the influence of poor model quality and slow model updates of some local agents on learning efficiency and accuracy. To solve this problem, some studies have applied asynchronous FL (AFL) frameworks to allow participating node selection in each global aggregation process [21], [22], [23]. Zhao et al. [21] formulated a joint resource allocation, data management, and user selection optimization problem to balance the tradeoff between model accuracy and FL costs in wireless networks. In [22], a big-data-enabled classification of encrypted mobile traffic was investigated through applying AFL to deep learning to improve time efficiency. A deep reinforcement learning (DRL)–based node selection algorithm was developed in [23] to improve efficiency and training accuracy in an AFL framework. In a distributed IoT environment, AFL frameworks have great potential to establish a global ML model with improved learning efficiency and model accuracy.

Using ML models to detect abnormal behaviors among IoT devices has attracted many research efforts with a view to securing critical infrastructure [10], [24], [25], [26], [27]. In [10], the authors designed an integrated model involving a convolutional neural network with long-short term memory for anomaly detection in an IoT time series. A one-class support Tucker machine method was designed in [24] for unsupervised outlier detection of IoT big data. However, the framework used in [10] and [24] relied on a central server to detect anomalies, which would cause network congestion and computing pressure for the central server. Some studies have proposed distributed frameworks to address these challenges, where FL is the prevailing technique [25], [26], [27]. In [25], the authors used clustering and classification methods to build a novel anomaly detection framework for hypertext transfer protocol, which drew on edge intelligence for IoT to distribute the entire detection process. To secure manufacturing processes for the industrial IoT, an FL-based distributed learning framework was proposed in [26] to combine all edge devices to train a global anomaly detection model. To address the efficiency, robustness, and security challenges facing FL, Cui et al. [27] proposed a differentially private AFL-based anomaly detection scheme for IoT infrastructure. However, in the literature on anomaly detection methods for IoT, there is a lack of discussion about the issue of continually moving sensing targets. Yet this is an issue that increases the difficulty of data sensing and collection.

UAVs and HAPSs can provide wireless sensing support from the sky to track moving sensing targets [15], [28], [29]. In [28], a cooperative Internet of UAVs was established to execute integrated sensing and transmission tasks for ground sensing targets. In [29], the authors proposed a joint sensing and transmission protocol to enable UAV-to-Device communication to improve UAV sensing services. Kurt and Yanikomeroglu [15] leveraged the sensing capability of a HAPS to serve autonomous devices for future aerial delivery networks. In addition to providing sensing capabilities, UAV-enabled and HAPS-enabled wireless technologies also play key roles in enhancing wireless connectivity and computation capability [30], [31], [32], [33]. Considering the constrained energy capacity in UAVs, [30], [31], [32] studied the UAV-enabled energy-efficient computation offloading, data uploading, and coverage enhancement systems, respectively. Compared to UAVs, HAPSs can provide wider coverage and stronger computational capabilities in a sustainable manner. Ren et al. [33] proposed a HAPS-assisted caching and computation offloading framework for intelligent transportation systems, where a HAPS played the dual role of a powerful computing server and a data library. Nevertheless, how to utilize the limited onboard energy storage in UAVs to complete different network tasks is still a vexing problem.

The secure information transmission between IoT devices and aerial networks is necessary to ensure data privacy and security. In [34], a secure identity-free transmission strategy was proposed to support UAV-aided IoT networks, where IoT devices uploaded their data packets to UAVs without location and identity information. Tang et al. [35] proposed a UAV-assisted data collection scheme for clustered IoT devices with a proof-of-stake blockchain-based security technique. In a VHetNet-enabled FL framework, a security strategy is needed to avoid parameter inference attacks. In [11], a privacy-preserving AFL framework for multi-UAV-enabled networks was developed for distributed ML model training, and a DRL-based algorithm was proposed to select high-quality devices to improve the efficiency. However, to the best of our knowledge, there are no studies that have considered anomaly detection and network scheduling jointly in a VHetNet-enabled FL framework, which is of great importance to achieve intelligent network management for 6G and beyond networks. We compare our work with the most relevant studies in Table 1.

### B. CONTRIBUTIONS AND ORGANIZATION

To tackle the aforementioned challenges, we propose a VHetNet-enabled AFL framework to implement the self-scheduling anomaly detection model, which can achieve high learning efficiency and model accuracy with the assistance of a network scheduling strategy. The proposed framework enables UAVs to train the anomaly detection model locally and upload model parameters to a HAPS for global aggregation without raw data transmission. The UAV selection process is introduced into the AFL framework to prevent UAVs with low local model quality and large energy consumption from affecting the learning efficiency and detection accuracy of the global model. To ensure the secure transmission between UAVs and the HAPS via wireless links, we add designed noise to local model parameters to achieve differential privacy (DP) during

**TABLE 1.** Comparison of our work with relevant studies.

| Related work | Device mobility | Network objective | Participating node selection in FL | Data transmission security | Resource limitation |
|---|---|---|---|---|---|
| [18] | ✓ | Network throughput, handoff cost | × | × | Bandwidth limitation |
| [21] | × | Model accuracy, cost | × | × | CPU and power constaints |
| [22] | × | Training time, cost | ✓ | ✓ | × |
| [23] | ✓ | Time cost, quality of learning | ✓ | ✓ | × |
| [25] | × | Speed and accuracy of anomaly detection | × | × | × |
| [26] | × | Accuracy of anomaly detection, communication overhead | × | × | × |
| [27] | × | Data utility | ✓ | ✓ | × |
| [28] | ✓ | Age of information | × | × | × |
| [11] | ✓ | Learning speed and accuracy | ✓ | × | Subchannel and computation capacity constraints |
| Our work | ✓ | Network coverage, learning speed and accuracy | ✓ | ✓ | UAV energy constraint |

the information exchange process. Considering the limited onboard energy storage and distinct model quality of UAVs, we also propose a compound-action actor-critic (CA2C)–based network scheduling algorithm to facilitate the efficient implementation of the self-scheduling anomaly detection model. More specifically, the main contributions of this paper are as follows:

- We develop a novel VHetNet-enabled AFL framework to provide an anomaly detection service for ubiquitous IoT. In the proposed framework, UAVs are responsible for sensing data from IoT devices and local model training, and the HAPS works as the control agent and aggregation node. UAVs with high local model quality and low energy consumption are selected to participate in the global aggregation instead of waiting for all UAVs to complete their local model update.
- A differentially private Wasserstein generative adversarial network with gradient penalty (WGAN-GP) is proposed as the basic anomaly detection model in UAVs, where DP is achieved by adding designed noise to gradients during the learning process.
- A joint device association, UAV selection, and UAV placement problem is formulated as a dynamic non-convex combination problem to maximize the number of sensed IoT devices and minimize the overall federated execution time and learning accuracy loss. To solve the dynamic problem, we introduce a CA2C-based network scheduling solution to determine both the discrete (including device association and UAV selection) and continuous (including UAV placement) actions, which will facilitate the efficient implementation of the self-scheduling anomaly detection model.
- To validate the efficiency and effectiveness of the proposed CA2C-AFL-based anomaly detection algorithm, we compare it with its three sub-algorithms, i.e., a deep deterministic policy gradient (DDPG)-FL algorithm, a deep Q-network (DQN)-AFL algorithm, and a standalone algorithm. Moreover, we introduce a centralized anomaly detection algorithm as the limit of performance. We conduct the simulation on both unlabeled and labeled real-world datasets. Simulation results demonstrate that the proposed algorithm can achieve superior detection accuracy with short federated execution time and low energy consumption.

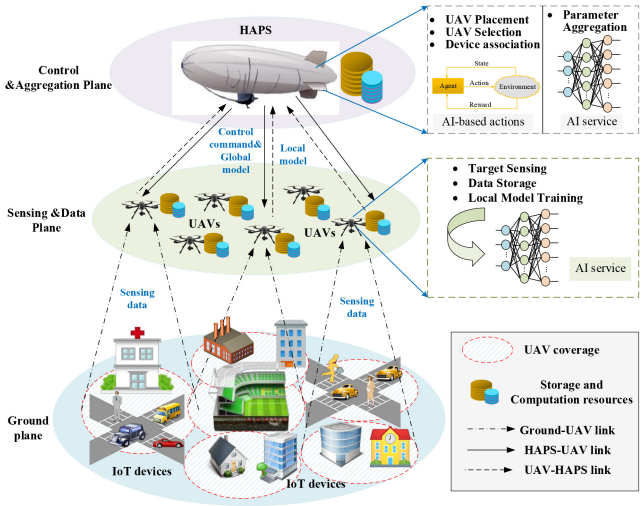



**FIGURE 1.** A three-layer VHetNet-enabled AFL framework.

The rest of the paper is organized as follows. Section II presents the system model and problem formulation of the VHetNet-enabled AFL-based anomaly detection framework. The CA2C-based network scheduling solution is proposed in Section III to facilitate the efficient implementation of the formulated framework. The performance of the proposed CA2C-AFL-based anomaly detection algorithm is evaluated in Section IV, and Section V concludes the paper.

## II. SYSTEM MODEL AND PROBLEM FORMULATION

Fig. 1 shows a three-layer VHetNet-enabled AFL framework, which consists of a HAPS, $N$ UAVs, and $K$ single-antenna IoT devices, denoted by $\mathcal{N} = \{1, \ldots, N\}$ and $\mathcal{K} = \{1, \ldots, K\}$, respectively. In the proposed framework, the main roles of each layer are elaborated as follows:

*Ground plane*. The ground plane consists of various IoT devices, such as monitoring sensors and control agents. Multiple IoT devices will produce large amounts of data, which can provide data support for the implementation of

the anomaly detection model. By collecting and analyzing the sensory data from IoT devices, the anomaly detection model can be constructed to detect abnormal behaviors and provide early warnings before the occurrence of failure.

*Sensing and data plane*. This plane is composed of UAVs. A UAV is responsible for sensing IoT devices within its coverage and training the local learning model using its sensory data. Using UAVs as aerial nodes to provide wireless sensing support from the sky is a promising paradigm due to their wider field of view, highly flexible and controllable 3D mobility, and the capability of sensing performance optimization through UAV position adjustment [16].

*Control and aggregation plane*. A HAPS in the control and aggregation plane works as the aerial center server. As a quasi-stationary network node, a HAPS operates in the stratosphere at an altitude of around 20 km [17]. Due to the advantages of its wide coverage and strong computational capabilities, a HAPS can control and manage the whole network intelligently using AI techniques. In the proposed framework, the HAPS is responsible for network control and the global aggregation of local models from UAVs. Compared to cloud servers, HAPSs can be deployed flexibly and powered by abundant energy sources, so HAPSs can work as the aerial center server efficiently. To decrease the single point of failure chances for the HAPS, regular backup of system data and configuration information is required in the HAPS, and the HAPS will coordinate with other neighboring HAPSs periodically.

### A. UAV SENSING

In our proposed VHetNet-enabled AFL framework, each UAV needs to sense data from its associated IoT devices and store the sensory data to update its local learning model periodically. A UAV is required to support the sensing and the local model update for $T$ time slots with its constrained energy. At time slot $t \in \mathcal{T} (\mathcal{T} = \{1, \ldots, T\})$, we denote the locations of device $k \in \mathcal{K}$ and UAV $n \in \mathcal{N}$ by $\boldsymbol{x}_k(t) = (x_k(t), y_k(t), 0)$ and $\boldsymbol{x}_n(t) = (x_n(t), y_n(t), z_n(t))$, respectively. We assume that the altitude $z_n(t)$ of each UAV $n$ does not change during any $T$ time slots, such that $z_n(t) \triangleq H, \ \forall n, t$. Hence, the distance between UAV $n$ and device $k$ is calculated by

$$d_{n,k}(t) = \sqrt{(x_n(t) - x_k(t))^2 + (y_n(t) - y_k(t))^2 + H^2}. \quad (1)$$

The successful sensing probability for IoT device $k$ by UAV $n$ can be expressed as an exponential function of the distance between them [14], [29], which is defined as

$$P_{n,k}(t) = e^{-\xi d_{n,k}(t)}, \quad \forall k, n, \quad (2)$$

where $\xi$ is a parameter reflecting the sensing performance. To count up the total number of sensed IoT devices, we set a minimum successful sensing probability threshold $P_{\text{th}}$ for UAVs. When UAV $n$ senses device $k$ at time slot $t$, the successful sensing probability should satisfy

$$P_{n,k}(t) \geq P_{\text{th}}, \quad \forall k, n, t. \quad (3)$$

We introduce the expression $\lambda_{n,k}(t)$ to denote whether or not device $k$ is associated with UAV $n$, with $\lambda_{n,k}(t) = 1$ indicating that device $k$ is associated with UAV $n$, and $\lambda_{n,k}(t) = 0$ indicating otherwise. If $\lambda_{n,k}(t) = 1$, then UAV $n$ is responsible for the sensing of device $k$ at time slot $t$. Hence, the total number of successfully sensed IoT devices at time slot $t$ is expressed as follows:

$$C^S(t) = \sum_{k=1}^{K} \sum_{n=1}^{N} \lambda_{n,k}(t) 1_{\{P_{n,k}(t) \geq P_{\text{th}}\}}. \quad (4)$$

To protect the data transmission process from IoT devices to UAVs, we introduce the secure identity-free transmission strategy proposed in [34] to transmit the sensing data and the location and identity information of IoT devices separately. By using this strategy, the sensing data of IoT devices are transmitted to UAVs for model training, and the location and identity information of IoT devices is uploaded to the HAPS for network control through different wireless links, so the privacy and anonymity of IoT devices are enhanced.

### B. AFL-BASED ANOMALY DETECTION MODEL

The occurrence of abnormal behaviors is rare, so the sensing data from IoT devices will form an imbalanced dataset. For this reason, we implement the anomaly detection model by first building a normal behavioral representation and then determining abnormal behaviors based on its deviation from the normal representation. In this paper, we use an improved generative adversarial network (GAN), namely WGAN-GP [36], as the basic anomaly detection model for its powerful ability to capture distribution from high-dimensional complex data. The AFL-based anomaly detection model assisted by WGAN-GP is illustrated in Fig. 2.

Similar to GAN, WGAN-GP consists of two models: generator $G$ and discriminator $D$. $G$ generates fake data $\bar{\boldsymbol{X}}$ using a random noise $\boldsymbol{z}$ that follows a known distribution $P_{\boldsymbol{z}}$, and $D$ attempts to distinguish $\bar{\boldsymbol{X}}$ from real data $\boldsymbol{X}$. $G$ and $D$ are trained through adversarial learning until $G$ can generate $\bar{\boldsymbol{X}}$ sharing the same distribution with $\boldsymbol{X}$ [37]. Let $P_{\boldsymbol{X}}$ and $P_{\bar{\boldsymbol{X}}}$ represent the distributions of real and fake data, and $D(\boldsymbol{X})$ denote the probability of $\boldsymbol{X}$ following the distribution $P_{\boldsymbol{X}}$. The training objective of WGAN-GP is given by

$$\min_G \max_D V(G, D) = \mathop{\mathbb{E}}_{\boldsymbol{X} \sim P_X} [D(\boldsymbol{X})] - \mathop{\mathbb{E}}_{\bar{\boldsymbol{X}} \sim P_{\bar{X}}} \left[D(\bar{\boldsymbol{X}})\right] + \eta \mathop{\mathbb{E}}_{\widehat{\boldsymbol{X}} \sim P_{\widehat{X}}} \left[\left(\left\|\nabla_{\widehat{X}} D(\widehat{\boldsymbol{X}})\right\|_2 - 1\right)^2\right], \quad (5)$$

where the last term represents a penalty on the gradient norm of $D$'s output with respect to random samples $\widehat{\boldsymbol{X}}$, which is defined as the weighted sum of $\boldsymbol{X}$ and $\bar{\boldsymbol{X}}$, and $\eta$ is the penalty coefficient. During the training process of WGAN-GP, $G$ and $D$ are updated alternately.

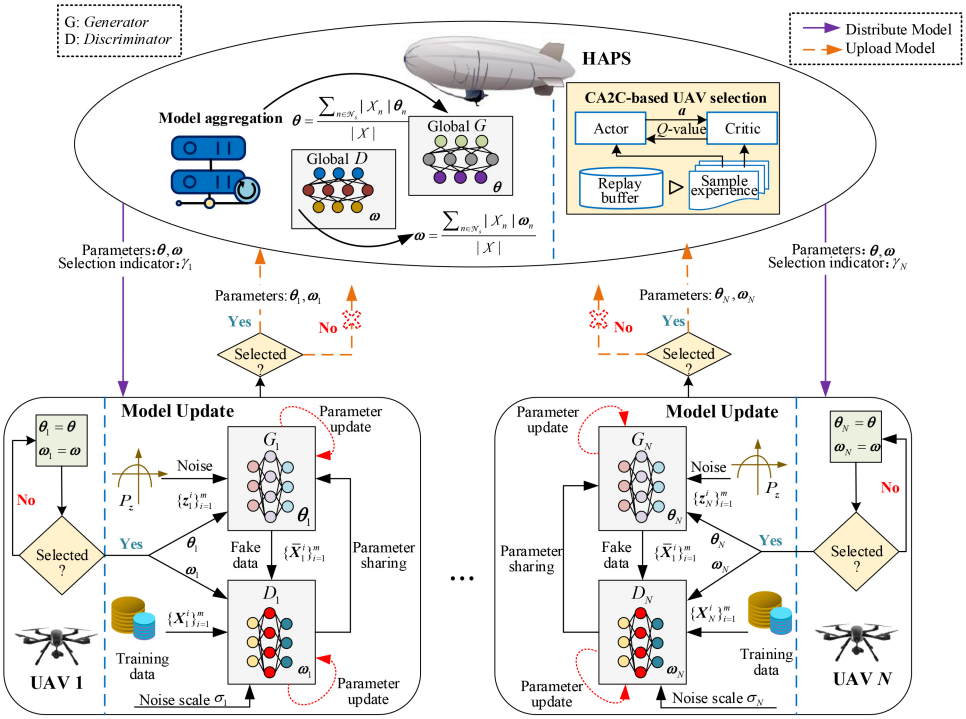



**FIGURE 2.** AFL-based anomaly detection framework assisted by differentially private WGAN-GP.

WGAN-GP is used as the local anomaly detection model in UAVs. First, we define the discriminator and generator parameters of UAV $n$ as $\boldsymbol{w}_n$ and $\boldsymbol{\theta}_n$, respectively. In each local update, UAV $n$ performs $N_d$ training iterations on discriminator $D_n$ between two generator training rounds. Assuming that $m$ is the batch size, we sample $m$ random noise $\{\boldsymbol{z}_n^i\}_{i=1}^m$ to generate fake data $\{\bar{\boldsymbol{X}}_n^i\}_{i=1}^m$, and sample $m$ real data $\{\boldsymbol{X}_n^i\}_{i=1}^m$ to train the local discriminator jointly. Hence, for UAV $n$, the loss function of $D_n$ is calculated by

$$LD_n(\boldsymbol{w}_n, \boldsymbol{\theta}_n) = -\frac{1}{m}\sum_{i=1}^{m}\left\{D\left(\boldsymbol{X}_n^i\right) - D\left(\bar{\boldsymbol{X}}_n^i\right) + \eta\left[\left\|\nabla_{\hat{\boldsymbol{X}}_n^i} D\left(\hat{\boldsymbol{X}}_n^i\right)\right\|_2 - 1\right]^2\right\}. \quad (6)$$

The proposed VHetNet-enabled AFL framework allows to execute local training in UAVs using their own sensory data and avoid raw data transmission to the HAPS for privacy preservation and communication efficiency. Meanwhile, the UAV selection strategy is introduced into the framework to keep UAVs with low local model quality and large energy consumption from affecting the learning efficiency and model accuracy. At the HAPS, the average loss function of the global discriminator is expressed as follows:

$$LD(\boldsymbol{w}, \boldsymbol{\theta}) = -\frac{1}{m|\mathcal{N}_s|}\sum_{n\in\mathcal{N}_s}\sum_{i=1}^{m}\left\{D\left(\boldsymbol{X}_n^i\right) - D\left(\bar{\boldsymbol{X}}_n^i\right) + \eta\left[\left\|\nabla_{\hat{\boldsymbol{X}}_n^i} D\left(\hat{\boldsymbol{X}}_n^i\right)\right\|_2 - 1\right]^2\right\}, \quad (7)$$

where $\boldsymbol{w}$ and $\boldsymbol{\theta}$ represent the global $D$ and $G$ parameters, respectively. $\mathcal{N}_S \subset \mathcal{N}$ is the selected UAV subset for global aggregation, and $|\mathcal{N}_s|$ is the number of selected UAVs. To avoid missing the important parts of training data stored in unselected UAVs, data that miss some training rounds will be tagged in these UAVs. When these UAVs return to better energy conditions or better model quality after a few idle periods or being recharged, these UAVs will join the global aggregation again, and the unused data will participate in the local training along with the new sensing data.

Similarly, the loss function of $G_n$ is calculated by

$$LG_n(\boldsymbol{w}_n, \boldsymbol{\theta}_n) = -\frac{1}{m}\sum_{i=1}^{m}\left[D\left(\bar{\boldsymbol{X}}_n^i\right)\right]. \quad (8)$$

Accordingly, at the HAPS, the average loss function of the global generator is expressed as

$$LG(\boldsymbol{w}, \boldsymbol{\theta}) = -\frac{1}{m|\mathcal{N}_s|}\sum_{n\in\mathcal{N}_s}\sum_{i=1}^{m}\left[D\left(\bar{\boldsymbol{X}}_n^i\right)\right]. \quad (9)$$

The AFL framework aims to search for the optimal model parameters that minimize the global loss accordingly:

$$\begin{aligned}\boldsymbol{w}^* &= \arg\min_{\boldsymbol{w}} LD(\boldsymbol{w}, \boldsymbol{\theta}) \\ \boldsymbol{\theta}^* &= \arg\min_{\boldsymbol{\theta}} LG(\boldsymbol{w}, \boldsymbol{\theta}).\end{aligned} \quad (10)$$

Although AFL has privacy advantages like FL, current research shows that sensitive information can still be inferred by using shared parameters [27]. To address this issue, we propose a differentially private WGAN-GP model, where DP

is achieved in WGAN-GP by adding designed noise to gradients during the learning process.

*Definition 1:* A randomized function $F$ is considered as $(\epsilon, \delta)$-differentially private if the following inequality is satisfied for any two databases $\boldsymbol{X}$ and $\boldsymbol{X}'$ differing in a single point and for any output subset $\boldsymbol{S}$ [38]:

$$\Pr(F(\boldsymbol{X}) \in \boldsymbol{S}) \leq e^{\epsilon} \cdot \Pr\big(F(\boldsymbol{X}') \in \boldsymbol{S}\big) + \delta, \tag{11}$$

where $F(\boldsymbol{X})$ and $F(\boldsymbol{X}')$ are the outputs of the function $F$ for inputs $\boldsymbol{X}$ and $\boldsymbol{X}'$, respectively.

Adding noise to the gradients of the Wasserstein distance is more efficient than adding noise to the final parameters directly with respect to preserving privacy [38]. The gradients $\Delta \boldsymbol{w}_n$ of discriminator parameter $\boldsymbol{w}_n$ after adding noise can be expressed as follows:

$$\Delta \boldsymbol{w}_n = -\frac{1}{m} \sum_{i=1}^{m} \Big\{ \nabla_{\boldsymbol{w}_n} LD_n^i + N\Big(0, \sigma_n^2 c_g^2 \boldsymbol{I}\Big) \Big\}, \tag{12}$$

where $\sigma_n$ represents the noise scale, and $c_g$ is the bound on gradients of the Wasserstein distance.

According to [38, Lemma 1], given the sampling rate $p = \frac{m}{M}$ (where $m$ is the batch size and $M$ is the total number of training data used in each discriminator iteration) and privacy violation $\delta$, for any positive $\epsilon$, the discriminator parameter guarantees $(\epsilon, \delta)$-DP with respect to all data used in the generator iteration if we choose

$$\sigma_n = 2p \sqrt{N_d \log\left(\frac{1}{\delta}\right)} \Big/ \epsilon. \tag{13}$$

The AFL-based differentially private WGAN-GP algorithm is summarized in Algorithm 1. The proposed algorithm can be extended to the dynamic dataset scenarios. Since UAVs collect data from IoT devices in real-time, when new sensing data accumulates to a certain scale, we continue to update the well-trained anomaly detection model with new datasets to adapt to dynamic dataset scenarios.

**Algorithm 1** AFL-Based Differentially Private WGAN-GP

**Input:** The number of time slots $T$; initial generator and discriminator parameters $\boldsymbol{\theta}$ and $\boldsymbol{w}$; the number of iterations $N_d$ between two generator training rounds; the number of local iterations $N_l$ per time slot; batch size $m$; Adam hyper-parameters $\alpha, \beta_1, \beta_2$; noise scale $\sigma_n$; bound on the gradient of Wasserstein distance $c_g$.

**Output:** Converged global model parameters $\boldsymbol{\theta}$ and $\boldsymbol{w}$.

**for** $t = 1 : T$ **do**
  Select the UAV subset $\mathcal{N}_s(t)$ by CA2C algorithm.
  For each UAV $n \in \mathcal{N}$, initialize local parameters by $\boldsymbol{\theta}_n = \boldsymbol{\theta}$, $\boldsymbol{w}_n = \boldsymbol{w}$.
  **for** $t_1 = 1 : N_l$ **do**
    **for** $t_2 = 1 : N_d$ **do**
      Sample a batch of noise $\{\boldsymbol{z}_n^i\}_{i=1}^m \sim P_{\boldsymbol{z}}$ and a batch of real data $\{\boldsymbol{X}_n^i\}_{i=1}^m \sim P_{\boldsymbol{X}}$.
      For each UAV $n \in \mathcal{N}_s(t)$, calculate $D_n$'s loss function $LD_n(\boldsymbol{w}_n, \boldsymbol{\theta}_n)$ according to (6).
      Add noise to the gradients $\Delta \boldsymbol{w}_n$ of discriminator parameter $\boldsymbol{w}_n$ according to (12).
      Update $\boldsymbol{w}_n$ using Adam optimizer: $\boldsymbol{w}_n \leftarrow \text{Adam}(\Delta \boldsymbol{w}_n, \boldsymbol{w}_n, \alpha, \beta_1, \beta_2)$.
    **end for**
    Sample another batch of noise $\{\boldsymbol{z}_n^i\}_{i=1}^m \sim P_{\boldsymbol{z}}$.
    For each UAV $n \in \mathcal{N}_s(t)$, calculate $G_n$'s loss function $LG_n(\boldsymbol{w}_n, \boldsymbol{\theta}_n)$ according to (8).
    Calculate the gradients of generator parameter $\Delta \boldsymbol{\theta}_n$ by $\Delta \boldsymbol{\theta}_n = \frac{1}{m} \sum_{i=1}^m \{\nabla_{\boldsymbol{\theta}_n} LG_n^i\}$.
    Update $\boldsymbol{\theta}_n$ using Adam optimizer: $\boldsymbol{\theta}_n \leftarrow \text{Adam}(\Delta \boldsymbol{\theta}_n, \boldsymbol{\theta}_n, \alpha, \beta_1, \beta_2)$.
  **end for**
  HAPS collects local parameters $\{\boldsymbol{w}_n\}_{n \in \mathcal{N}_s(t)}$ and $\{\boldsymbol{\theta}_n\}_{n \in \mathcal{N}_s(t)}$ from selected UAVs, and updates the global parameters $\boldsymbol{w} = \sum_{n \in \mathcal{N}_s(t)} |\mathcal{X}_n| \boldsymbol{w}_n / |\mathcal{X}|$, $\boldsymbol{\theta} = \sum_{n \in \mathcal{N}_s(t)} |\mathcal{X}_n| \boldsymbol{\theta}_n / |\mathcal{X}|$, where $|\mathcal{X}_n|$ is the cardinality of training set $\mathcal{X}_n$ and $|\mathcal{X}| = \sum_{n \in \mathcal{N}_s(t)} |\mathcal{X}_n|$.
**end for**

### C. COMMUNICATION MODEL

The information exchange between UAVs and the HAPS includes the processes of UAVs uploading local models to the HAPS for global aggregation and the HAPS distributing the global model to all UAVs after the averaging operation. We consider line-of-sight (LoS) links for both UAV-HAPS and HAPS-UAV wireless transmission. The LoS path loss in dB between UAV $n$ and the HAPS is modeled by

$$l_n^{\text{LoS}}(t) = 20 \log(4\pi d_n(t) f_c / c) + h^{\text{LoS}}, \tag{14}$$

where $f_c$ is the carrier frequency, and $d_n(t)$ is the distance between UAV $n$ and the HAPS at time $t$. $c$ denotes the speed of light, and $h^{\text{LoS}}$ is the mean additional loss of LoS links caused by the free space propagation [11]. So the uplink transmission rate from UAV $n$ to the HAPS is given by

$$R_n^u(t) = B^u \log_2\left(1 + \frac{P_n^u 10^{-l_n^{\text{LoS}}(t)/10}}{\Gamma B^u N_0}\right), \tag{15}$$

where $\Gamma > 1$ accounts for the gap from the channel capacity due to the practical modulation and coding scheme used [39], $B^u$ is the uplink channel bandwidth, which is assumed to be the same for all UAVs. $P_n^u$ is the transmission power of UAV $n$, and $N_0$ is the Gaussian noise power spectrum density.

Similarly, the downlink transmission rate from the HAPS to UAV $n$ is calculated by

$$R_n^d(t) = B^d \log_2\left(1 + \frac{P_n^d 10^{-l_n^{\text{LoS}}(t)/10}}{\Gamma B^d N_0}\right), \tag{16}$$

where $B^d$ is the downlink channel bandwidth, which is also assumed to be the same for all UAVs. $P_n^d$ is the transmission power that HAPS allocates to UAV $n$.

### D. AFL MODEL UPDATE LATENCY

Due to the different quality of local models and different battery power remaining in different UAVs, we try to select a subset of UAVs in each training round to participate in the global aggregation with the goal of minimizing the federated execution time and learning accuracy loss. We introduce $\gamma_n(t)$ to denote whether or not UAV $n$ is selected for global

aggregation at time slot $t$, with $\gamma_n(t) = 1$ indicating that UAV $n$ is selected, and $\gamma_n(t) = 0$ indicating otherwise.

The one-round federated execution time includes local model update and upload latency, and global model aggregation and distribution latency.

*1) Local model update latency.* We define $\xi_n^a(t)$ to represent the computation capability of UAV $n$ to execute training tasks at time slot $t$, and $\rho_n$ represents the number of CPU cycles needed to train the local model on one unit of data. Therefore, the local model update latency of UAV $n$ at time slot $t$ is given by

$$T_n^{\text{update}}(t) = \frac{|\mathcal{X}_n(t)|\rho_n}{\xi_n^a(t)}, \tag{17}$$

where $|\mathcal{X}_n(t)|$ is the cardinality of dataset $\mathcal{X}_n(t)$.

*2) Local model upload latency.* Let $L(|\boldsymbol{\theta}_n| + |\boldsymbol{w}_n|)$ denote the total number of bits required by UAV $n$ to upload its local parameters to the HAPS. So for UAV $n$, the local model upload latency is expressed as

$$T_n^{\text{upload}}(t) = \frac{L(|\boldsymbol{\theta}_n| + |\boldsymbol{w}_n|)}{R_n^u(t)}. \tag{18}$$

*3) Global model aggregation latency.* We define $\mathcal{N}_s(t) = \{n|\gamma_n(t) = 1, \forall n\}$ to represent the UAV subset that is selected for global aggregation at time $t$. Assuming that the global model aggregation latency is linearly related to the number of participating UAVs, which is given by

$$T_n^{\text{aggregation}}(t) = \varsigma|\mathcal{N}_s(t)|, \tag{19}$$

where $|\mathcal{N}_s(t)|$ represents the number of selected UAVs, and $\varsigma$ denotes the unit time for global aggregation.

*4) Global model distribution latency.* Let $L(|\boldsymbol{\theta}| + |\boldsymbol{w}|)$ denote the total number of bits required by the HAPS to distribute the global model to UAVs. So the global model distribution latency for UAV $n$ is expressed as

$$T_n^{\text{distribution}}(t) = \frac{L(|\boldsymbol{\theta}| + |\boldsymbol{w}|)}{R_n^d(t)}. \tag{20}$$

The time cost for AFL is expressed as the maximum one-round execution time in $\mathcal{N}_s(t)$, which is given by

$$T(t) = \max_{n\in\mathcal{N}_s(t)} \Big(T_n^{\text{update}}(t) + T_n^{\text{upload}}(t) + T^{\text{aggregation}}(t) + T_n^{\text{distribution}}(t)\Big). \tag{21}$$

Hence, we define the time cost function as

$$C^T(t) = \frac{1}{|\mathcal{N}_s(t)|} \sum_{n\in\mathcal{N}_s(t)} \Big(T_n^{\text{update}}(t) + T_n^{\text{upload}}(t) + T^{\text{aggregation}}(t) + T_n^{\text{distribution}}(t)\Big). \tag{22}$$

### E. UAV ENERGY MODEL

UAV energy consumption mainly consists of three important parts: propulsion energy, computational energy, and transmission energy.

*1) Propulsion energy.* According to [14] and [39], the propulsion energy of UAV $n$ during a flight is modeled by

$$E_n^P(t) = \underbrace{\frac{\|\boldsymbol{x}_n(t) - \boldsymbol{x}_n(t-1)\|}{V_n}}_{\text{flying time}} \times \underbrace{\left[\kappa_1 V_n^3 + \frac{\kappa_2}{V_n} + \kappa_3\left(1 + \frac{V_n^2}{g^2}\right)\right]}_{\text{propulsion power}}, \tag{23}$$

where $\kappa_1, \kappa_2, \kappa_3$, and $g$ are constants related to the type of UAVs, and $V_n$ denotes the flying velocity of UAV $n$.

*2) Computational energy.* We define $P_n^c$ as the computational power of UAV $n$. Accordingly, the computational energy consumption of UAV $n$ at time slot $t$ is calculated by

$$E_n^C(t) = \gamma_n(t) P_n^c T_n^{\text{update}}(t). \tag{24}$$

*3) Transmission energy.* As $P_n^u$ is the transmission power of UAV $n$, the transmission energy consumption of UAV $n$ at time slot $t$ is given by

$$E_n^M(t) = \gamma_n(t) P_n^u T_n^{\text{upload}}(t). \tag{25}$$

The onboard energy storage of each UAV is limited and represented by $E_n^{\max}$. To support the system for $T$ time slots, the total energy consumption of UAV $n$ should satisfy

$$\sum_{t=1}^{T} E_n^P(t) + E_n^C(t) + E_n^M(t) \le E_n^{\max}. \tag{26}$$

### F. PROBLEM FORMULATION

In the three-layer VHetNet, we use UAVs to sense data from IoT devices and employ an AFL framework to establish a global anomaly detection model without local data centralization at the HAPS, which can enhance the data privacy and communication efficiency. For this AFL framework, a UAV selection strategy is desirable to select UAVs with high local model quality and low energy consumption to participate in the global aggregation at the HAPS. In addition, it is necessary to schedule locations of UAVs and device association to provide the best coverage for IoT devices with minimum energy consumption. Overall, our objective is to maximize the total number of sensed IoT devices by UAVs and minimize AFL model execution time and learning accuracy loss over all time slots, which leads to an efficient anomaly detection model. Therefore, we formulate the optimization problem as follows:

$$\begin{aligned}
\min_{\substack{\{\boldsymbol{x}_n(t)\},\{\lambda_{n,k}(t)\}\\ \{\gamma_n(t)\},\{\boldsymbol{w}(t)\}}} \ & \frac{1}{T}\sum_{t=1}^{T}\Big(-\mu_1 C^S(t) + \mu_2 C^T(t) + \mu_3 LD(\boldsymbol{w}(t), \boldsymbol{\theta}(t))\Big) \\
s.t. \ \ & \text{a) } \boldsymbol{\theta}(t) = \arg\min_{\boldsymbol{\theta}(t)} LG(\boldsymbol{w}(t), \boldsymbol{\theta}(t)) \\
& \text{b) } \lambda_{n,k}(t) \in \{0,1\}, \ \gamma_n(t) \in \{0,1\}, \ \forall n,k,t \\
& \text{c) } \sum_{n=1}^{N} \lambda_{n,k}(t) \le 1, \ \forall k,t \\
& \text{d) } \sum_{t=1}^{T} E_n^P(t) + E_n^C(t) + E_n^M(t) \le E_n^{\max}, \ \forall n,
\end{aligned} \tag{27}$$

where $\mu_1$, $\mu_2$, and $\mu_3$ are constant weight parameters for the total number of sensed IoT devices $C^S(t)$, the execution time $C^T(t)$, and the learning accuracy loss $LD(\boldsymbol{w}(t), \boldsymbol{\theta}(t))$. In this system, the learning accuracy losses $LD(\boldsymbol{w}(t), \boldsymbol{\theta}(t))$ and $LG(\boldsymbol{w}(t), \boldsymbol{\theta}(t))$ are measured at the end of each time slot. Constraint (c) indicates that each IoT device can only associate with one UAV to perform sensing tasks. Constraint (d) is used to limit the maximum energy consumption.

The optimization problem (27) is challenging to solve because it is a non-convex combination and NP-hard problem. In addition, due to the time-varying feature of IoT device locations and UAV battery power, it is difficult for traditional optimization algorithms to address this problem. Model-free reinforcement learning (RL) is a promising dynamic programming technique that is capable of handling sequential decision-making processes in dynamic environments [40], [41]. Therefore, we introduce RL to solve this problem.

## III. CA2C-BASED NETWORK SCHEDULING SOLUTION

### A. MODELING OF RL ENVIRONMENT

We model the sequential decision-making problem (27) as a Markov decision process (MDP) represented by $<\mathcal{S}, \mathcal{A}, \mathcal{P}, r>$, where $\mathcal{S}$, $\mathcal{A}$, $\mathcal{P}$, and $r$ denote state space, action space, state transition function, and reward, respectively. In the three-layer VHetNet, the HAPS is responsible for observing the dynamic environment and tries to maximize the expected cumulative reward. RL is used to tackle the joint UAV selection, device association, and UAV placement problem for assisting the implementation of the self-scheduling anomaly detection. According to the system model, four elements in the MDP are defined as follows.

*State:* At each time slot $t$, the network state $\boldsymbol{s}(t) \in \mathcal{S}$ consists of current locations of IoT devices $\{\boldsymbol{x}_k(t)\}_{k\in\mathcal{K}}$, UAV locations $\{\boldsymbol{x}_n(t-1)\}_{n\in\mathcal{N}}$ at last time slot $t-1$, and remaining energy of UAVs $\{E_n^{\mathrm{Re}}(t)\}_{n\in\mathcal{N}}$. Therefore, the network state at time slot $t$ is expressed as

$$\boldsymbol{s}(t) = \{\{\boldsymbol{x}_k(t)\}_{k\in\mathcal{K}}, \{\boldsymbol{x}_n(t-1)\}_{n\in\mathcal{N}}, \{E_n^{\mathrm{Re}}(t)\}_{n\in\mathcal{N}}\}. \quad (28)$$

*Action:* At time slot $t$, the HAPS is responsible for selecting an action $\boldsymbol{a}(t) \in \mathcal{A}$ based on the observed state $\boldsymbol{s}(t)$. $\boldsymbol{a}(t)$ consists of UAV locations $\{\boldsymbol{x}_n(t)\}_{n\in\mathcal{N}}$, device association indicators $\{\lambda_{n,k}(t)\}_{k\in\mathcal{K},n\in\mathcal{N}}$, and UAV selection indicators $\{\gamma_n(t)\}_{n\in\mathcal{N}}$, which is expressed as

$$\boldsymbol{a}(t) = \{\boldsymbol{x}_n(t)\}_{n\in\mathcal{N}}, \{\lambda_{n,k}(t)\}_{k\in\mathcal{K},n\in\mathcal{N}}, \{\gamma_n(t)\}_{n\in\mathcal{N}}\}. \quad (29)$$

*State transition function:* Let $\mathcal{P}(\boldsymbol{s}(t+1)|\boldsymbol{s}(t), \boldsymbol{a}(t))$ represent the transition probability of the network environment from the current state $\boldsymbol{s}(t)$ to a new state $\boldsymbol{s}(t+1)$ after taking an action $\boldsymbol{a}(t)$.

*Reward:* Aiming to maximize the total number of sensed IoT devices while minimizing the federated execution time and learning accuracy loss under energy constraints, the instant reward $r(t)$ is required to evaluate the quality of policy $\boldsymbol{\pi}$ under the current state-action pair $(\boldsymbol{s}(t), \boldsymbol{a}(t))$ [11]. To obtain the optimal policy $\boldsymbol{\pi}^*$ under energy constraints, we adopt the ReLu function $f(k) = \max(k, 0)$ to calculate the variable $\Theta_n(t)$ with respect to the constraint 27(d) as

$$\Theta_n(t) = \max\left((E_n^P(t) + E_n^C(t) + E_n^M(t)) - \frac{E_n^{\max}}{T}, 0\right), \ \forall n. \quad (30)$$

Therefore, the instant reward function is defined as

$$r(t) = -(C(t) + \nu\|\boldsymbol{\Theta}(t)\|_1), \quad (31)$$

where $C(t) = -\mu_1 C^S(t) + \mu_2 C^T(t) + \mu_3 LD(\boldsymbol{w}(t), \boldsymbol{\theta}(t))$ is the total weighted cost. $\boldsymbol{\Theta}(t)$ denotes the penalty function, which is the vector of $\Theta_n(t)$. $||\cdot||_1$ is the L1-norm operator.

The objective for the MDP model is to find a policy $\boldsymbol{\pi}$ mapping the state $\boldsymbol{s}(t)$ to action $\boldsymbol{a}(t)$, i.e., $\boldsymbol{a}(t) = \boldsymbol{\pi}(\boldsymbol{s}(t))$, which is capable of maximizing the total accumulative reward denoted by

$$R(t) = \sum_{t'=t}^{T} \chi^{t'-t} r(t'), \quad (32)$$

where $\chi \in (0, 1)$ is a discount factor indicating the impact of future rewards on current reward.

In the three-layer VHetNet, since both state and action spaces are large, DRL algorithms that combine deep neural networks (DNNs) and RL are more effective for solving the large-scale decision-making problems. However, existing DRL algorithms cannot be applied directly to solve the combinatorial optimization problem (27). This is because that actions taken in problem (27) involve both continuous UAV position variables $\{\boldsymbol{x}_n(t)\}_{n\in\mathcal{N}}$ as well as discrete device association $\{\lambda_{n,k}(t)\}_{k\in\mathcal{K},n\in\mathcal{N}}$ and UAV selection indicators $\{\gamma_n(t)\}_{n\in\mathcal{N}}$, whereas existing DRL algorithms are suitable for problems with purely continuous or purely discrete action spaces.

### B. CA2C ALGORITHM

To overcome this challenge, we introduce the CA2C algorithm proposed by [28], which combines the advantages of DDPG for dealing with continuous decision variables and DQN for dealing with discrete decision variables. The training and implementation processes of the CA2C-based network scheduling algorithm are illustrated in Fig. 3.

*DQN:* In DQN, DNNs are utilized to approximate the Q-function, which is denoted by $Q(\boldsymbol{s}(t), \boldsymbol{a}(t); \boldsymbol{\vartheta})$, where $\boldsymbol{\vartheta}$ denotes DNN parameters. We train the deep Q-function to achieve the best fitting by minimizing the loss function $L(\boldsymbol{\vartheta})$ in each iteration, which is defined as the expectation of mean squared error between the estimated Q-value and target value, given by $L(\boldsymbol{\vartheta}) = \mathbb{E}[(y(t) - Q(\boldsymbol{s}(t), \boldsymbol{a}(t); \boldsymbol{\vartheta}))^2]$. In $L(\boldsymbol{\vartheta})$, $y(t) = r(t) + \chi \max_{\boldsymbol{a}'} Q(\boldsymbol{s}(t+1), \boldsymbol{a}'; \boldsymbol{\vartheta})$ is the target value, and $\boldsymbol{a}'$ is the action generated by $\varepsilon$-greedy to strike a balance between exploration and exploitation. By $\varepsilon$-greedy, the agent selects a random action $\boldsymbol{a}' \in \mathcal{A}$ with probability $\varepsilon$ and selects the best action that follows the greedy policy $\boldsymbol{a}' = \arg\max_{\boldsymbol{a}'} Q(\boldsymbol{s}(t+1), \boldsymbol{a}'; \boldsymbol{\vartheta})$ with probability $1 - \varepsilon$. Despite the accurate approximation, DNNs may cause divergence

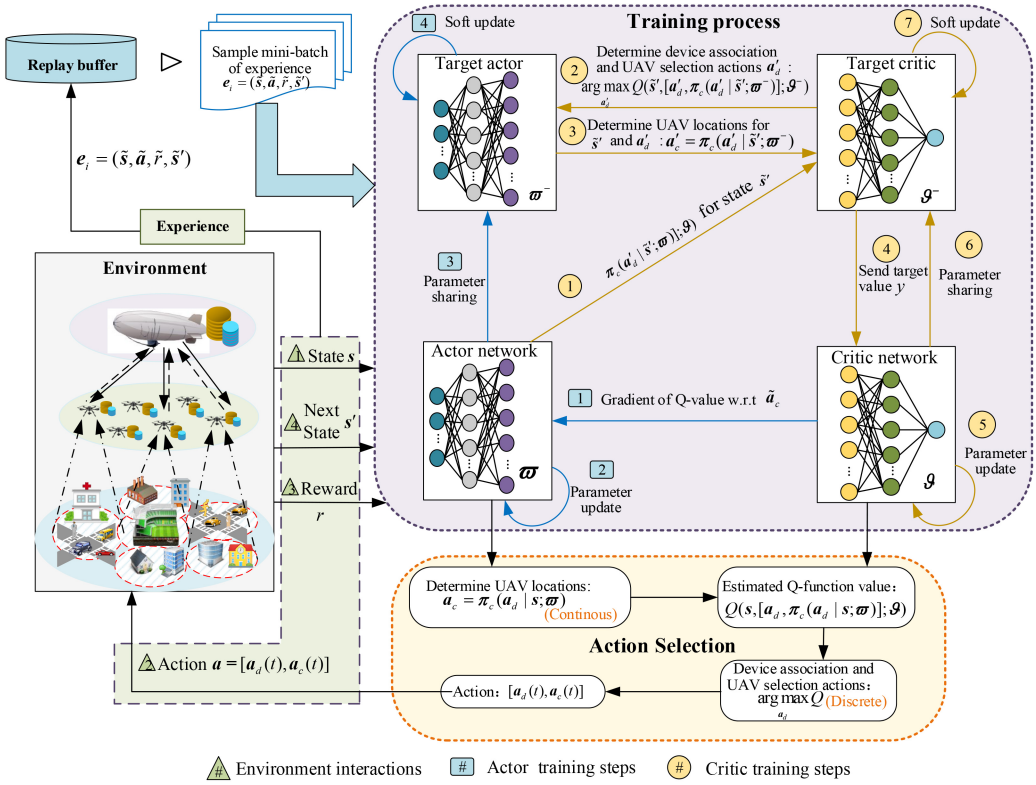



**FIGURE 3.** Training and implementation processes of the CA2C-based network scheduling algorithm.

or ineffective learning due to non-stationary target values and the correlation among samples. To overcome these difficulties, a pair of techniques are introduced in DQN, namely fixed target network and experience replay [42]. Hence, the loss function is re-written as

$$L(\boldsymbol{\vartheta}) = \mathbb{E}_U\big[(r(t) + \chi \max_{\boldsymbol{a}'} Q(s(t+1), \boldsymbol{a}'; \boldsymbol{\vartheta}^-) - Q(\boldsymbol{s}(t), \boldsymbol{a}(t); \boldsymbol{\vartheta}))^2\big], \tag{33}$$

where $\boldsymbol{\vartheta}^-$ represents parameters of the target network. At the beginning of the training process, $\boldsymbol{\vartheta}$ and $\boldsymbol{\vartheta}^-$ are initialized with the same values. However, $\boldsymbol{\vartheta}^-$ only updates with $\boldsymbol{\vartheta}$ every $N_u$ steps and keeps unchanged between two individual $\boldsymbol{\vartheta}$ updates (slower than updates of $\boldsymbol{\vartheta}$) to avoid a divergence in the training process. $U$ represents the replay memory, and random mini-batches are uniformly sampled from $U$ when performing updates to break the correlation among samples. DQN is only suitable for environments with discrete action spaces. Since problem (27) includes both continuous actions (UAV placement) and discrete actions (device association and UAV selection indicators), it is difficult to solve the problem optimally with the DQN method.

*DDPG:* Policy gradient-based RL methods can be used to handle sequential decision-making problems with continuous action spaces. These methods aim to optimize a policy using gradients of the expected reward. To speed up the convergence of policy gradient-based methods, DDPG has been proposed to combine the policy-based and value-based methods to estimate the policy gradient more efficiently [43]. In DDPG, there are two different neural networks, an actor and a critic, with parameters $\boldsymbol{\varpi}$ and $\boldsymbol{\vartheta}$, respectively. For state $\boldsymbol{s}(t)$, the actor obtains the continuous action $\boldsymbol{a}(t)$ based on a deterministic policy $\boldsymbol{\pi}(\boldsymbol{s}(t); \boldsymbol{\varpi})$, and the critic evaluates the quality of actions taken via the Q-function $Q(\boldsymbol{s}(t), \boldsymbol{a}(t); \boldsymbol{\vartheta})$. Both the deterministic policy $\boldsymbol{\pi}(\boldsymbol{s}(t); \boldsymbol{\varpi})$ and the Q-function $Q(\boldsymbol{s}(t), \boldsymbol{a}(t); \boldsymbol{\vartheta})$ are approximated by DNNs. DDPG also uses the target network and experience replay to facilitate the training process. Sampling random mini-batches from experience replay $U$, the critic is updated by minimizing its loss function $L(\boldsymbol{\vartheta})$, which is expressed as the expectation of a mean squared error between the estimated value $Q(\boldsymbol{s}(t), \boldsymbol{a}(t); \boldsymbol{\vartheta})$ and the target value $y(t) = r(t) + \chi Q(\boldsymbol{s}(t+1), \boldsymbol{\pi}(\boldsymbol{s}(t+1); \boldsymbol{\varpi}^-); \boldsymbol{\vartheta}^-)$, given by

$$L(\boldsymbol{\vartheta}) = \mathbb{E}_U\Big[(y(t) - Q(\boldsymbol{s}(t), \boldsymbol{a}(t); \boldsymbol{\vartheta}))^2\Big], \tag{34}$$

where $\boldsymbol{\varpi}^-$ and $\boldsymbol{\vartheta}^-$ represent parameters of the target actor and the target critic, respectively. The actor is updated using the policy gradient method as follows [43]:

$$\nabla_{\boldsymbol{\varpi}} J(\boldsymbol{\varpi}) = \mathbb{E}_U[\nabla_{\boldsymbol{\varpi}} \pi(\boldsymbol{s}(t); \boldsymbol{\varpi}) \nabla_{\boldsymbol{a}} Q(s(t), \boldsymbol{a}(t); \boldsymbol{\vartheta})]. \tag{35}$$

*CA2C:* To solve problem (27), which includes both continuous and discrete actions, we decompose the optimal policy $\boldsymbol{\pi}^*$ into two parts, i.e., the policy for finding the optimal continuous UAV positions $\boldsymbol{\pi}_c^*(\boldsymbol{a}_d(t)|\boldsymbol{s}(t); \boldsymbol{\varpi})$, and the policy for selecting the optimal discrete device association and UAV

**Algorithm 2** Training Processes for Critic

**Input:** Sample $I$ experience samples $\boldsymbol{e}_i = (\tilde{\boldsymbol{s}}, \tilde{\boldsymbol{a}}, \tilde{r}, \tilde{\boldsymbol{s}}')|_{i=1}^{I}$ from replay buffer $U$; critic parameter $\boldsymbol{\vartheta}$; target critic parameter $\boldsymbol{\vartheta}^-$; soft update parameter $\tau$.
**Output:** Updated parameters $\boldsymbol{\vartheta}$ and $\boldsymbol{\vartheta}^-$.
1: Obtain the UAV location $\boldsymbol{\pi}_c(\tilde{\boldsymbol{a}}_d|\tilde{\boldsymbol{s}}'; \boldsymbol{\varpi})$ for state $\tilde{\boldsymbol{s}}'$.
2: Determine device association and UAV selection actions $\boldsymbol{a}_d'$ for state $\tilde{\boldsymbol{s}}'$ based on the Q-value estimated by the target critic.
3: Target actor calculates the UAV location $\boldsymbol{a}_c'$ for state $\tilde{\boldsymbol{s}}'$.
4: According to (34), calculate $y$ by adding $\tilde{r}$ and the output Q-value $Q(\tilde{\boldsymbol{s}}', [\boldsymbol{a}_d', \boldsymbol{\pi}_c(\boldsymbol{a}_d'|\tilde{\boldsymbol{s}}'; \boldsymbol{\varpi}^-)]; \boldsymbol{\vartheta}^-)$ of target critic.
5: Update $\boldsymbol{\vartheta}$ by using the Adam optimizer in the critic.
6: Update $\boldsymbol{\vartheta}^-$ in the target critic via soft update $\boldsymbol{\vartheta}^- = (1-\tau)\boldsymbol{\vartheta}^- + \tau\boldsymbol{\vartheta}$.

**Algorithm 3** Training Processes for Actor

**Input:** Sample $I$ experience samples $\boldsymbol{e}_i = (\tilde{\boldsymbol{s}}, \tilde{\boldsymbol{a}}, \tilde{r}, \tilde{\boldsymbol{s}}')|_{i=1}^{I}$ from replay buffer $U$; actor parameter $\boldsymbol{\varpi}$; target actor parameter $\boldsymbol{\varpi}^-$; soft update parameter $\tau$.
**Output:** Updated parameters $\boldsymbol{\varpi}$ and $\boldsymbol{\varpi}^-$.
1: According to (37), calculate the gradients of Q-value $Q(\tilde{\boldsymbol{s}}, [\tilde{\boldsymbol{a}}_d, \tilde{\boldsymbol{a}}_c]; \boldsymbol{\vartheta})$ with respect to the UAV locations for all the sampled experiences.
2: Update $\boldsymbol{\varpi}$ by using the Adam optimizer in the actor.
3: Update $\boldsymbol{\varpi}^-$ in the target actor via soft update $\boldsymbol{\varpi}^- = (1-\tau)\boldsymbol{\varpi}^- + \tau\boldsymbol{\varpi}$.

selection indicators $\boldsymbol{a}_d^*(t)$ given the optimal UAV positions. More specifically, the optimal discrete policy of device association and UAV selection $\boldsymbol{a}_d^*(t)$, which maximizes the Q-value for the network state $\boldsymbol{s}(t)$, is determined by

$$\boldsymbol{a}_d^*(t) = \arg\max_{\boldsymbol{a}_d(t)} Q^*\big(\boldsymbol{s}(t), \big[\boldsymbol{a}_d(t), \boldsymbol{\pi}_c^*(\boldsymbol{a}_d(t)|\boldsymbol{s}(t); \boldsymbol{\varpi})\big]; \boldsymbol{\vartheta}\big), \tag{36}$$

where policy $\boldsymbol{\pi}_c^*(\boldsymbol{a}_d(t)|\boldsymbol{s}(t); \boldsymbol{\varpi})$ provides the optimal UAV positions given state $\boldsymbol{s}(t)$ and discrete policy $\boldsymbol{a}_d^*(t)$. DNNs are adopted to approximate functions $Q^*$ and $\boldsymbol{\pi}_c^*$ in view of large state and action spaces. Moreover, the training of DNNs in CA2C is executed by combining training methods used in DQN and DDPG.

Algorithm 2 and 3 summarize the training processes of the actor and critic, respectively. The loss function of the actor is $J(\boldsymbol{\varpi}(t)) = \mathbb{E}_U[(Q(\boldsymbol{s}(t), [\boldsymbol{a}_d(t), \boldsymbol{\pi}_c(\boldsymbol{a}_d(t)|\boldsymbol{s}(t); \boldsymbol{\varpi})]; \boldsymbol{\vartheta})]$. The actor is updated by using the policy gradient method as

$$\nabla_{\boldsymbol{\varpi}} J(\boldsymbol{\varpi}) = \mathbb{E}_U[\nabla_{\boldsymbol{\varpi}} \boldsymbol{\pi}_c(\boldsymbol{a}_d(t)|\boldsymbol{s}(t); \boldsymbol{\varpi}) \times \nabla_{\boldsymbol{a}_c} Q(\boldsymbol{s}(t), [\boldsymbol{a}_d(t), \boldsymbol{a}_c(t)]; \boldsymbol{\vartheta})|_{\boldsymbol{a}_c(t)=\pi_c(\boldsymbol{a}_d(t)|\boldsymbol{s}(t);\boldsymbol{\varpi})}]. \tag{37}$$

The corresponding loss function of the critic network is expressed as follows:

$$\begin{aligned} L(\boldsymbol{\vartheta}) &= \mathbb{E}_U\Big[(y(t) - Q(\boldsymbol{s}(t), \boldsymbol{a}(t); \boldsymbol{\vartheta}))^2\Big] \\ y(t) &= r(t) + Q(\boldsymbol{s}(t+1), \big[\boldsymbol{a}_d', \boldsymbol{\pi}_c\big(\boldsymbol{a}_d'|\boldsymbol{s}(t+1); \boldsymbol{\varpi}^-\big)\big]; \boldsymbol{\vartheta}^-) \\ \boldsymbol{a}_d' &= \arg\max_{\boldsymbol{a}_d'} Q(\boldsymbol{s}(t+1), \big[\boldsymbol{a}_d', \boldsymbol{\pi}_c(\boldsymbol{a}_d'|\boldsymbol{s}(t+1); \boldsymbol{\varpi})\big]; \boldsymbol{\vartheta}). \end{aligned} \tag{38}$$

**Algorithm 4** CA2C-Based Network Scheduling Algorithm

**Input:** Initial parameters $\boldsymbol{\vartheta}$, $\boldsymbol{\varpi}$, $\boldsymbol{\vartheta}^-$, $\boldsymbol{\varpi}^-$; replay buffer $U = \emptyset$; exploration parameter $\varepsilon$; the maximum number of training episodes $N_{\text{ep}}$; the number of time slots $T$; mini-batch size $I$.
**Output:** Converged model parameters $\boldsymbol{\vartheta}$ and $\boldsymbol{\varpi}$.
1: **for** *episode* $= 1 : N_{\text{ep}}$ **do**
2: Receive initial observation state $\boldsymbol{s}$.
3: **for** $t = 1 : T$ **do**
4: With probability $\varepsilon$, choose random device association and UAV selection actions $\boldsymbol{a}_d(t)$, otherwise choose $\boldsymbol{a}_d(t)$ according to (36).
5: Determine the UAV locations for the selected device association and UAV selection actions as $\boldsymbol{a}_c(t) = \boldsymbol{\pi}_c(\boldsymbol{a}_d(t)|\boldsymbol{s}(t); \boldsymbol{\varpi})$.
6: Transition to a new state $\boldsymbol{s}(t+1)$ and get instant reward $r(t)$.
7: Store experience $\boldsymbol{s}(t)$, $\boldsymbol{a}(t) = [\boldsymbol{a}_d(t), \boldsymbol{a}_c(t)]$, $r(t)$ and $\boldsymbol{s}(t+1)$ in replay buffer $U$ as $\boldsymbol{e}_i = (\tilde{\boldsymbol{s}}, \tilde{\boldsymbol{a}}, \tilde{r}, \tilde{\boldsymbol{s}}')$.
8: Sample $I$ experience samples $\boldsymbol{e}_i = (\tilde{\boldsymbol{s}}, \tilde{\boldsymbol{a}}, \tilde{r}, \tilde{\boldsymbol{s}}')|_{i=1}^{I}$ from replay buffer $U$.
9: Train the critic according to Algorithm 2.
10: Train the actor according to Algorithm 3.
11: **end for**
12: **end for**

In (38), we choose an action for the next state according to the trained networks (i.e., the actor and the critic, with parameters $\boldsymbol{\varpi}$ and $\boldsymbol{\vartheta}$, respectively), while we estimate the Q-value for the next state using target networks (i.e., the target actor and the target critic, with parameters $\boldsymbol{\varpi}^-$ and $\boldsymbol{\vartheta}^-$, respectively) [28]. The detailed steps of the CA2C algorithm are presented in Algorithm 4.

*Computational complexity:* The HAPS is responsible for the action selection, training processes of the CA2C algorithm, and the global aggregation of local WGAN-GP models from UAVs. According to [28], the computational complexity of calculating the output given an input is proportional to the sum of input and output sizes of DNNs. Based on the state and action spaces defined in MDP, the sizes of inputs in actor and critic networks are $4N+2K+1+NK$ and $6N+2K+1+NK$, respectively. Meanwhile, the sizes of outputs in actor and critic networks are $2N$ and 1, respectively. So the computational complexity of determining UAV locations and of estimating the Q value for a state-action pair is $O(NK)$. Since the HAPS needs to estimate Q values of all discrete actions, the computational complexity for the action selection is $O(N^2K^2)$. The computational complexity of the model training is proportional to the product of input and output sizes of DNNs [28]. So the computational complexity for training processes is $O(IN^2K^2)$, where $I$ represents the size of training batch. The computational complexity for the global aggregation is $O(N)$.

UAVs are responsible for the training of local WGAN-GP models. In the WGAN-GP model, the sizes of inputs in generator and discriminator networks are $\dim(\boldsymbol{X})$ and $\dim(\boldsymbol{X}) + \dim(\boldsymbol{z})$, respectively, where $\dim(\boldsymbol{X})$ and $\dim(\boldsymbol{z})$ represent the dimensionality of real data and random noise, respectively. Meanwhile, the sizes of outputs in generator

and discriminator networks are dim($\boldsymbol{z}$) and 1, respectively. Since the discriminator will train $N_d$ times between two generator training rounds, the computational complexity for the training of local WGAN-GP model is $O(N_d m \cdot \dim(\boldsymbol{z}) \cdot \dim(\boldsymbol{X}))$, where $m$ represents the size of training batch..

## IV. SIMULATION RESULTS AND ANALYSIS

In this section, we conduct simulations to evaluate the performance of the proposed self-scheduling anomaly detection scheme for ubiquitous IoT. In doing so, the proposed CA2C-AFL algorithm is implemented in pytorch1.11 (Python 3.7) and carried out by a computer with a CPU capacity of 12 Intel Core i7-10750H CPU 2.6 GHz and a RAM of 16 GB. We compare the proposed algorithm with the following four algorithms in terms of the detection performance, the energy consumption, and the execution time:

*1) DQN-AFL:* This algorithm uses DQN to implement the discrete device association and UAV selection strategies, and the continuous UAV placement is implemented in a random manner, so we can evaluate the effect of UAV placement strategy by comparing it with the proposed algorithm.

*2) DDPG-FL:* This algorithm adopts DDPG to learn the continuous UAV placement strategy, and the device association is based on the minimum distance. No UAV selection process is included, so all UAVs participate in the training of the FL-based anomaly detection model. By comparing it with the proposed algorithm, we can evaluate the effect of including and excluding the UAV selection strategy.

*3) Standalone:* In the standalone algorithm, the continuous UAV placement is implemented in a random manner, the device association is based on the minimum distance, and no UAV selection process is included. Moreover, each UAV trains its own anomaly detection model without any data or information exchange, so we can evaluate the effect of AFL framework by comparing it with the proposed algorithm.

*3) Centralized:* In the centralized algorithm, the HAPS is responsible for collecting sensing data from all UAVs, and trains the centralized WGAN-GP model for anomaly detection. The CA2C is also trained in HAPS for learning joint device association and UAV placement strategies. The centralized algorithm is taken as the limit of performance.

Note that the DQN-AFL, the DDPG-FL, and the standalone algorithms also use WGAN-GP as the local anomaly detection model in UAVs.

### A. EXPERIMENT SCENARIO

For our simulations, we consider a VHetNet that consists of a HAPS and five airborne UAVs to support a coverage area of 1 km $\times$ 1 km, where 20 target devices are randomly located. The sensing parameter $\xi$ and the minimum successful sensing probability $P_{\text{th}}$ of a UAV are set as 0.0001 and 0.99, respectively. In this case, the maximum sensing distance of a UAV is 100 m. the The transmission power of the HAPS and each UAV is set as 33 dBm and 26 dBm, respectively. The total energy storage of each UAV is 50 kJ and the computation capability of each UAV is assumed to be 80,000 cycles/s. The bandwidth of the uplink and downlink between the HAPS and each UAV is set as 5 MHZ and 20 MHZ, respectively.

The size of the transmitted data for the model parameters is 5 kbits, and the unit computation power is set as 5 W. The batch size for the CA2C training is assumed to be 256. Other relevant simulation parameters are listed in Table 2, where the UAV setting parameters follow the UAV energy model established in [39].

**TABLE 2. Simulation parameters.**

| Parameters | Description | Value |
|---|---|---|
| $f_c$ | Carrier frequency | 2 GHz |
| $c$ | Speed of light | $3 \times 10^8$ m/s |
| $h^{\text{LoS}}$ | Mean additional loss for LoS links | 10 dB |
| $\Gamma$ | The gap index from the channel capacity | 2 |
| $N_0$ | Gaussian noise power spectrum density | -174 dBm/Hz |
| $m$ | Batch size for WGAN-GP training | 20 |
| $L(\lvert\boldsymbol{\theta}_n\rvert + \lvert\boldsymbol{w}_n\rvert)$ | The number of bits to upload/distribute parameters | 100 kbits |
| $\rho_n$ | The number of CPU cycles needed for training the local model on one batch data | 4 cycles/s |
| $p$ | The sampling rate ($\frac{m}{M}$) | $\frac{20}{24000}$ |
| $\delta$ | The privacy violation probability | $10^{-5}$ |
| $\epsilon$ | The privacy level | 8 |
| $N_d$ | The number of discriminator updates between two generator iterations | 6 |
| $N_l$ | The number of local iterations per time slot | 20 |
| $\kappa_1, \kappa_2, \kappa_3, g$ | UAV type parameters | 0.009, 357, 80, 69 |
| $V_n$ | UAV flying velocity | 30 m/s |
| $H$ | The altitude of UAVs | 50 m |
| $\mu_1, \mu_2, \mu_3$ | Constant weight parameters | 2, 4, 10 |
| $\nu$ | Penalty coefficient for energy | 0.01 |
| $\chi$ | Discount factor | 0.9 |
| $\varepsilon$ | Probability for exploration in greedy rule | 0.1 |
| $U$ | The capacity of replay buffer | 2000 |
| $T$ | The number of time slots in each episode | 50 |
| $N_{\text{ep}}$ | Maximum number of training episodes | 200 |

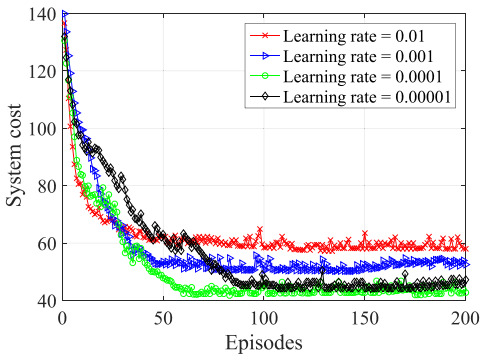
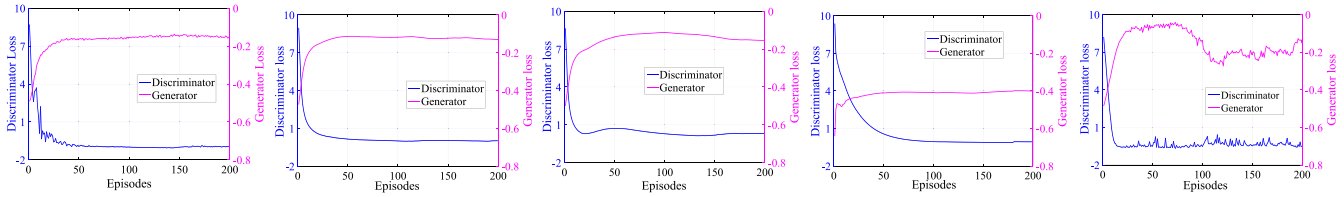



**FIGURE 4. Convergence of the CA2C approach in terms of system cost with different learning rates.**




(a) Centralized (b) CA2C-AFL (c) DQN-AFL (d) DDPG-FL (e) Standalone

**FIGURE 5.** Convergence of the differentially private WGAN-GP model under different training frameworks.

*Datasets*: We choose an unlabeled dataset and a labeled dataset as sensing data from ground IoT devices to validate the effectiveness of the proposed algorithm. The unlabeled dataset is the LabData published by Intel Berkeley Research Lab [44], which includes temperature, humidity, light, and voltage features collected from 54 distributed Mica2Dot sensors. The labeled dataset is the X-IIoTID published on IEEE Dataport platform, which includes 31 network traffic features collected from 13 industrial IoT service types with normal and attack (abnormal) labels [45]. Each UAV dataset is built on the sensing data from its associated IoT devices. We divide each UAV subset into three disjoint datasets: a training dataset for model training, a validation dataset for setting the discriminant criterion of anomalies, and a test dataset to evaluate the detection performance [37].

### B. CONVERGENCE COMPARISON

We first evaluate the convergence performance of the proposed CA2C approach with different learning rates, which are set as 0.01, 0.001, 0.0001, and 0.00001, sequentially. Fig. 4 shows the convergence of the system cost during the training processes of the CA2C approach. The system cost is the objective function defined in (27), which is the weighted combination of the total number of sensed IoT devices, the execution time, and the learning accuracy loss. Note that an episode includes 50 time slots. Learning rates have a major impact on system cost and convergence speed. From Fig. 4, we can see that although a larger learning rate will mean a faster convergence process, it will lead to a higher system cost. However, the smaller learning rate does not always lead to less system cost. As shown in Fig. 4, when the learning rate is smaller than 0.0001 (i.e., equal to 0.00001), the convergence process becomes slower, but the achieved system cost is not smaller than the situation when the learning rate is set as 0.0001. Therefore, we set the learning rate as 0.0001 to account for both the convergence speed and system cost.

Fig. 5 compares the convergence of the proposed differentially private WGAN-GP-based anomaly detection model under five different training frameworks, which are built on the centralized, the proposed CA2C-AFL, the DQN-AFL, the DDPG-FL, and the standalone approaches, respectively. From Fig. 5, we can see that both the discriminator and generator losses can converge after certain episodes, which indicates that the generator and discriminator can be successfully trained in all five training frameworks. However, under the standalone training framework, the convergence speed is slow, and the training process is hard to converge to a stable value. This is because the standalone algorithm trains the proposed model in each UAV without any information exchange, which causes poor data abundance and model robustness. Because the proposed CA2C-AFL algorithm will determine the best UAV placement strategy and select the best UAV subset with high model quality in the training process, it achieves the best convergence performance compared to the other four training frameworks.

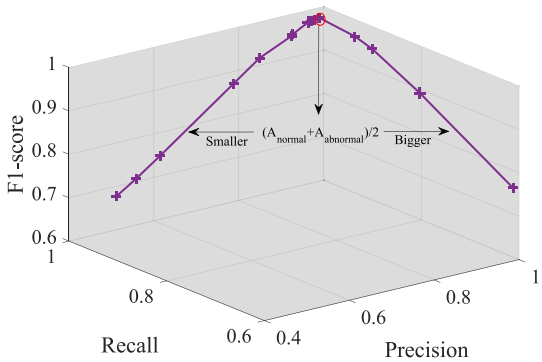



**FIGURE 6.** The relationship among precision, recall, and F1-score.

### C. ANOMALY DETECTION PERFORMANCE

There are three basic evaluation metrics used to reflect the performance of different anomaly detection algorithms, namely precision, recall, and F1-score, whose meanings are explained as follows:

*Precision*: The ratio of correctly identified abnormal behaviors relative to all behaviors identified as abnormal.

*Recall*: The ratio of correctly identified abnormal behaviors relative to all behaviors set as abnormal in advance.

*F1-score*: The weighted average of precision and recall, given by $F1\text{-}score = 2 \times (precision \times recall)/(precision + recall)$.

Once the proposed CA2C-AFL-based anomaly detection model is trained well, it can be used to identify abnormal behaviors by calculating their anomaly scores, which are defined as the weighted combination of the discriminator loss and generator loss [37]. If the anomaly score of new data is greater than a preset threshold, it will be determined as abnormal. The threshold values have a major impact on

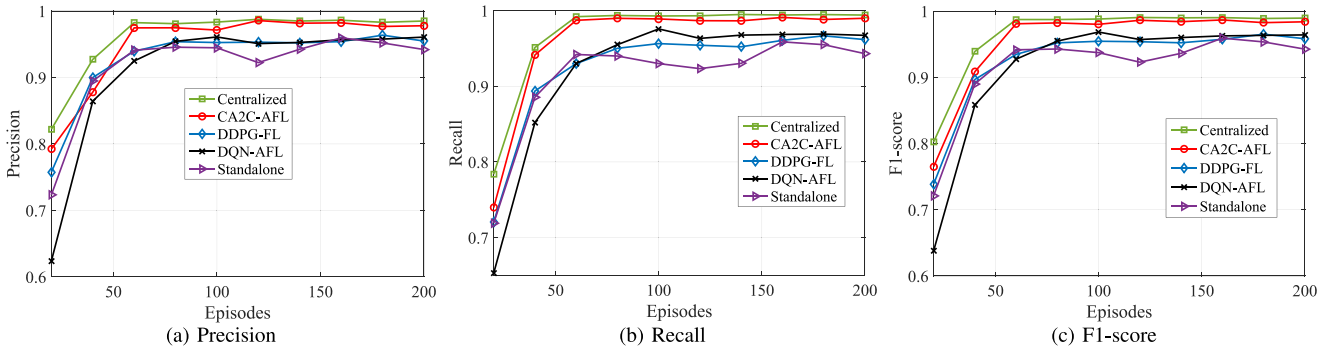


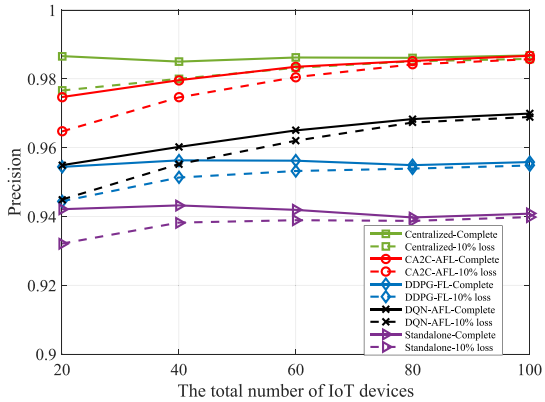

**FIGURE 7.** Detection performance comparison among five different algorithms.

the performance of the proposed anomaly detection model. We obtain the threshold by using the validation dataset. First, to simulate the abnormal behaviors in the unlabeled dataset, we inject the noise factor to samples in the validation dataset with a probability 0.5, where the noise factor follows the Gaussian distribution $N(\mu, \sigma^2)$ with the mean $\mu = 0.1$ and the variance $\sigma^2 = 0.1$. Then, we calculate the average anomaly scores $A_{\text{normal}}$ and $A_{\text{abnormal}}$ of the normal and abnormal data in the validation dataset. Finally, we choose different threshold values in range $[A_{\text{normal}}, A_{\text{abnormal}}]$ to capture the relationship among precision, recall, and F1-score, which is shown in Fig. 6. As we can see, when we set the threshold as $(A_{\text{normal}} + A_{\text{abnormal}})/2$, precision, recall, and F1-score metrics all achieve their best values. Therefore, the threshold for the anomaly score is defined as $(A_{\text{normal}} + A_{\text{abnormal}})/2$ to achieve the best detection performance.

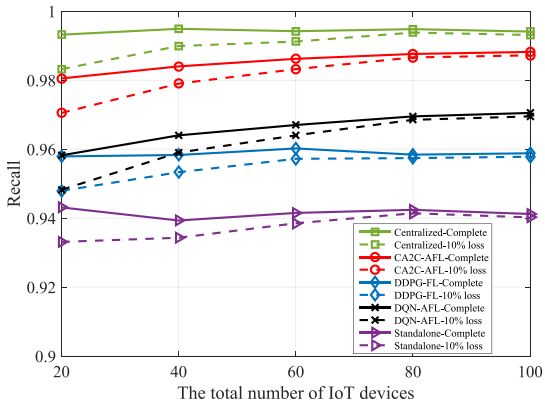
Fig. 7 shows the detection performance of five different algorithms: the centralized, the proposed CA2C-AFL, the DDPG-FL, the DQN-AFL, and the standalone anomaly detection algorithms. The test dataset is used to evaluate the performance of different algorithms, and precision, recall, and F1-score are used as the evaluation metrics. For the unlabeled dataset, we also inject the noise factor to samples in the test dataset to simulate the abnormal behaviors, where the noise factor also follows the Gaussian distribution $N(0.1, 0.1)$. We can see that the convergence trends of the three evaluation metrics are nearly consistent with those of discriminator and generator losses, which are shown in Fig. 5. It indicates that the five anomaly detection algorithms can capture the accurate data distribution from the training dataset with enough training episodes. By contrast, the proposed CA2C-AFL algorithm is superior to the DDPG-FL, the DQN-AFL, and the standalone algorithms, and slightly inferior to the centralized algorithm in terms of all three evaluation metrics. The detection performance of the DDPG-FL and DQN-AFL algorithms is inferior to our proposed algorithm because these two include either the UAV placement strategy or the UAV selection strategy, but not both as our algorithm does. Besides, the detection performance of the standalone algorithm is the most unstable and inaccurate due to the lack of information fusion and exchange.

Fig. 8 presents the detection performance achieved by five different algorithms with different numbers of IoT devices. The number of IoT devices changes from 20, 40, 60, 80 to 100, and the number of UAVs changes from 5, 10, 15, 20 to 25, accordingly. There exist data losses in UAV sensing due to the mobility of IoT devices and UAVs. To evaluate the influence of sensing data losses on the anomaly detection model, we have further compared the detection performance of five different algorithms between models trained with the complete dataset and models trained with the incomplete dataset that has 10% random data losses. From Fig. 8, we can see that the centralized algorithm can achieve the best detection performance, and the proposed CA2C-AFL algorithm and the DQN-AFL algorithm have better detection performance than the DDPG-FL and standalone algorithms under different numbers of IoT devices. The superior detection performance of CA2C-AFL and DQN-AFL algorithms benefit from their UAV selection processes, which prevent UAVs with low model quality from affecting the overall accuracy. The precision, recall, and F1-score values achieved by CA2C-AFL and DQN-AFL algorithms increase with the increase of IoT devices, but the detection performance achieved by the other three algorithms without UAV selection processes maintains at a stable level with some fluctuations when the number of IoT devices increases. Moreover, we can see that the influence of sensing data losses will become smaller with the increase of IoT devices, and when the total number of IoT devices reaches 80, anomaly detection models trained with the complete dataset and the incomplete dataset can achieve nearly the same recall, precision and F1-score values. Therefore, to reduce the influence of sensing data losses on model accuracy, improving the network coverage to connect with more IoT devices is necessary.

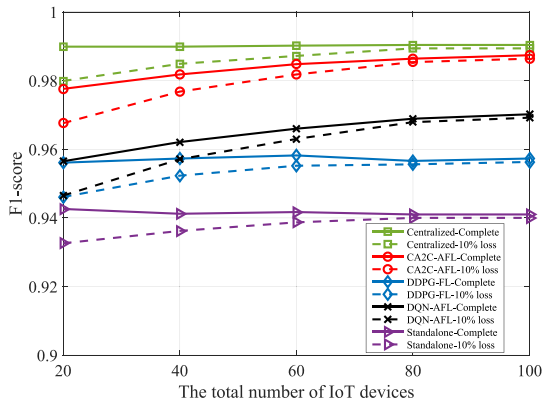
To evaluate the effect of adding noise on the detection performance, we compare the performance of the CA2C-AFL algorithm without DP with the proposed CA2C-AFL algorithm with DP. The six different algorithms mentioned above are evaluated with both the unlabeled and labeled




(a) Precision




(b) Recall




(c) F1-score

**FIGURE 8.** Detection performance comparison with different numbers of IoT devices.

datasets, and the comparison results are summarized in Table 3. By contrast, the detection performance of the centralized algorithm performs best among the six algorithms in terms of both the unlabeled and labeled datasets. Although the performance of the proposed CA2C-AFL algorithm is degraded by adding noise, it is not much different from the CA2C-AFL algorithm without DP. It proves that we can use

**TABLE 3.** Detection performance comparisons.

| Algorithms | Unlabeled dataset (LabData) | | | Labeled dataset (X-IIoTID) | | |
|---|---|---|---|---|---|---|
| | Precision | Recall | F1-score | Precision | Recall | F1-score |
| CA2C-AFL | 0.9779 | 0.9899 | 0.9839 | 0.9823 | 0.9910 | 0.9866 |
| CA2C-AFL (without DP) | 0.9786 | 0.9898 | 0.9842 | 0.9826 | 0.9911 | 0.9868 |
| DDPG-FL | 0.9549 | 0.9617 | 0.9583 | 0.9607 | 0.9633 | 0.9620 |
| DQN-AFL | 0.9608 | 0.9673 | 0.9641 | 0.9704 | 0.9726 | 0.9715 |
| Standalone | 0.9421 | 0.9431 | 0.9426 | 0.9503 | 0.9525 | 0.9514 |
| Centralized | 0.9850 | 0.9941 | 0.9895 | 0.9884 | 0.9948 | 0.9916 |

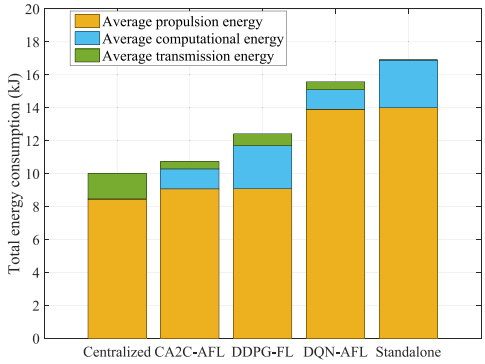



**FIGURE 9.** Comparison of average UAV energy consumption among five different algorithms.

the DP method to further protect the data privacy, and there is little degradation in the detection performance.

### D. ENERGY CONSUMPTION AND LEARNING DELAY

We also compare the UAV energy consumption of the proposed CA2C-AFL algorithm with the centralized, DDPG-FL, DQN-AFL, and standalone algorithms. Fig. 9 shows the average UAV energy consumption of different activities, including propulsion, computation and transmission. The energy values presented in Fig. 9 are averaged over five UAVs, then further averaged over the total 200 episodes. From Fig. 9, we can see that the centralized algorithm consumes the least UAV energy among the five algorithms, because it centralizes the training of the anomaly detection model to the HAPS, and the computational energy is excluded from the UAV energy consumption. The proposed CA2C-AFL algorithm consumes less energy than the other three algorithms. The reason is that the proposed CA2C-AFL algorithm will determine the optimal locations of UAVs with the aim of saving energy, so it consumes less propulsion energy than the DQN-AFL and standalone algorithms. Besides, unlike FL, in AFL, just a subset of UAVs need to update and upload their local models in one global training round, so the proposed CA2C-AFL algorithm consumes less transmission and computational energy than the DDPG-FL algorithms. The standalone algorithm does not need to consume energy in transmission activities, but its computational energy is the highest among the five algorithms.

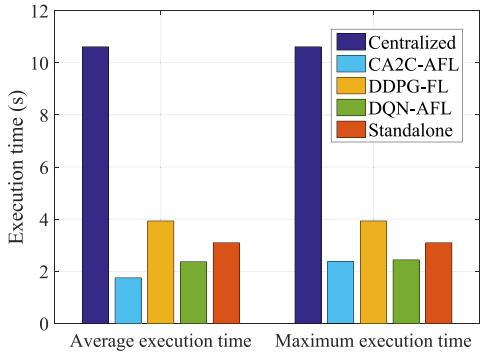



**FIGURE 10.** Comparisons of execution time for anomaly detection model training.

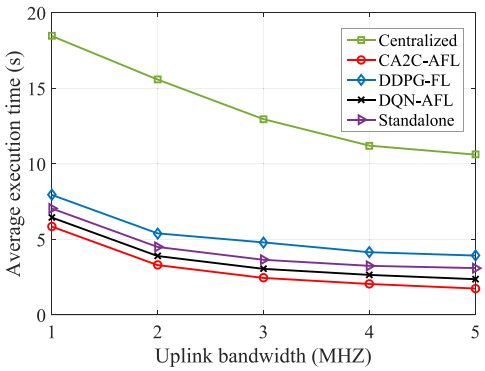



**FIGURE 11.** Comparisons of average execution time with different uplink bandwidths.

Fig. 10 shows the average execution time and maximum execution time of the five different algorithms over the total 200 episodes. The execution time is measured by the consumed time for training within all 50 time slots. By contrast, the centralized algorithm consumes the longest execution time because it requires the HAPS to collect sensing data from all UAVs for the centralized processing. The DDPG-FL algorithm needs longer execution time than the other three algorithms because it needs to wait for all UAVs to complete their local model updates before the global aggregation process. The proposed CA2C-AFL algorithm has the shortest execution time. Note that the average execution time of the proposed algorithm and the DQN-AFL algorithm is less than their maximum execution time due to the presence of the UAV subset selection process.

The data transmission rates from UAVs to the HAPS will greatly influence the system performance considered in this paper, especially the federated execution time. To show the impact of data transmission rates on the federated execution time, we change the uplink bandwidth from 1 MHZ to 5 MHZ with the interval of 1 MHZ. Fig. 11 shows the average execution time of five algorithms with different transmission rates. From Fig. 11, we can see that the average execution time will decrease with the increase of data transmission rates. Moreover, the proposed CA2C-AFL algorithm can achieve the shortest execution time under different transmission rates.

## V. CONCLUSION AND FUTURE DIRECTIONS

In this paper, we studied a VHetNet-enabled AFL-based anomaly detection framework for ubiquitous IoT devices with the assistance of a network scheduling strategy, which aimed to improve learning efficiency and detection accuracy. More specifically, the framework was designed to train local anomaly detection models at UAVs based on their sensory data and aggregated local models at a HAPS in an asynchronous manner. This aimed to decrease the federated learning execution time as well as the computation and transmission overheads. To ensure the secure transmission between UAVs and the HAPS, we adopted a differentially private WGAN-GP as the local anomaly detection model. Moreover, considering the limited onboard energy storage of UAVs, we formulated a joint device association, UAV selection, and UAV placement problem, which we solved using the CA2C approach to facilitate the efficient implementation of the self-scheduling anomaly detection model. Simulation results validated the effectiveness of the proposed framework in terms of efficiency and accuracy.

The present study motivates several future research directions: (i) the privacy protection and secure transmission mechanism in wireless networks when using distributed learning frameworks; (ii) the online learning of ML model to adapt to dynamic dataset scenarios; (iii) the integration of communication, sensing and computing capability for the efficient and intelligent service provision, such as anomaly detection, demand prediction and function orchestration for network devices; (iv) the synergy of space-air-ground integrated networks and AI techniques to extend the intelligence to everywhere and everything.

## REFERENCES

[1] W. Wu et al., "AI-native network slicing for 6G networks," *IEEE Wireless Commun.*, vol. 29, no. 1, pp. 96–103, Feb. 2022.
[2] Z. Wang, Z. Zhou, H. Zhang, G. Zhang, H. Ding, and A. Farouk, "AI-based cloud-edge-device collaboration in 6G space-air-ground integrated power IoT," *IEEE Wireless Commun.*, vol. 29, no. 1, pp. 16–23, Feb. 2022.
[3] T. Darwish, G. K. Kurt, H. Yanikomeroglu, G. Senarath, and P. Zhu, "A vision of self-evolving network management for future intelligent vertical HetNet," *IEEE Wireless Commun.*, vol. 28, no. 4, pp. 96–105, Aug. 2021.
[4] A. Yu, Q. Yang, L. Dou, and M. Cheriet, "Federated imitation learning: A cross-domain knowledge sharing framework for traffic scheduling in 6G ubiquitous IoT," *IEEE Netw.*, vol. 35, no. 5, pp. 136–142, Sep./Oct. 2021.
[5] K. B. Letaief, W. Chen, Y. Shi, J. Zhang, and Y.-J. A. Zhang, "The roadmap to 6G: AI empowered wireless networks," *IEEE Commun. Mag.*, vol. 57, no. 8, pp. 84–90, Aug. 2019.
[6] X.-X. Lin, P. Lin, and E.-H. Yeh, "Anomaly detection/prediction for the Internet of Things: State of the art and the future," *IEEE Netw.*, vol. 35, no. 1, pp. 212–218, Jan./Feb. 2021.
[7] J. Jiang, G. Han, L. Liu, L. Shu, and M. Guizani, "Outlier detection approaches based on machine learning in the Internet-of-Things," *IEEE Wireless Commun.*, vol. 27, no. 3, pp. 53–59, Jun. 2020.
[8] F. Li, A. Shinde, Y. Shi, J. Ye, X.-Y. Li, and W. Song, "System statistics learning-based IoT security: Feasibility and suitability," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6396–6403, Aug. 2019.
[9] D. Wu, Z. Jiang, X. Xie, X. Wei, W. Yu, and R. Li, "LSTM learning with bayesian and Gaussian processing for anomaly detection in industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 16, no. 8, pp. 5244–5253, Aug. 2020.

[10] 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.*, vol. 52, no. 1, pp. 112–122, Jan. 2022.

[11] H. Yang, J. Zhao, Z. Xiong, K.-Y. Lam, S. Sun, and L. Xiao, "Privacy-preserving federated learning for UAV-enabled networks: Learning-based joint scheduling and resource management," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 10, pp. 3144–3159, Oct. 2021.

[12] D. Van Huynh, T. D.-Duy, L. D. Nguyen, M.-T. Le, N.-S. Vo, and T. Q. Duong, "Real-time Optimized path planning and energy consumption for data collection in unmanned ariel vehicles-aided intelligent wireless sensing," *IEEE Trans. Ind. Informat.*, vol. 18, no. 4, pp. 2753–2761, Apr. 2022.

[13] Y. Li et al., "Data collection maximization in IoT-sensor networks via an energy-constrained UAV," *IEEE Trans. Mobile Comput.*, vol. 22, no. 1, pp. 159–174, Jan. 2023.

[14] S. Zhang, H. Zhang, Z. Han, H. V. Poor, and L. Song, "Age of information in a cellular Internet of UAVs: Sensing and communication trade-off design," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6578–6592, Oct. 2020.

[15] G. K. Kurt and H. Yanikomeroglu, "Communication, computing, caching, and sensing for next-generation aerial delivery networks: Using a high-altitude platform station as an enabling technology," *IEEE Veh. Technol. Mag.*, vol. 16, no. 3, pp. 108–117, Sep. 2021.

[16] Q. Wu et al., "A comprehensive overview on 5G-and-beyond networks with UAVs: From communications to sensing and intelligence," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 10, pp. 2912–2945, Oct. 2021.

[17] G. K. Kurt et al., "A vision and framework for the high altitude platform station (HAPS) networks of the future," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 729–779, 2nd Quart., 2021.

[18] Y.-J. Liu, G. Feng, Y. Sun, S. Qin, and Y.-C. Liang, "Device association for RAN slicing based on hybrid federated deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 15731–15745, Dec. 2020.

[19] X. Huang, S. Leng, S. Maharjan, and Y. Zhang, "Multi-agent deep reinforcement learning for computation offloading and interference coordination in small cell networks," *IEEE Trans. Veh. Technol.*, vol. 70, no. 9, pp. 9282–9293, Sep. 2021.

[20] X. Wang, C. Wang, X. Li, V. C. M. Leung, and T. Taleb, "Federated deep reinforcement learning for Internet of Things with decentralized cooperative edge caching," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9441–9455, Oct. 2020.

[21] Z. Zhao et al., "Federated learning with non-IID data in wireless networks," *IEEE Trans. Wireless Commun.*, vol. 21, no. 3, pp. 1927–1942, Mar. 2022.

[22] G. Aceto, D. Ciuonzo, A. Montieri, V. Persico, and A. Pescap, "Know your big data trade-offs when classifying encrypted mobile traffic with deep learning," in *Proc. Netw. Traffic Meas. Anal. Conf. (TMA)*, 2019, pp. 121–128.

[23] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, "Blockchain empowered asynchronous federated learning for secure data sharing in Internet of Vehicles," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4298–4311, Apr. 2020.

[24] X. Deng, P. Jiang, X. Peng, and C. Mi, "An intelligent outlier detection method with one class support tucker machine and genetic algorithm toward big sensor data in Internet of Things," *IEEE Trans. Ind. Electron.*, vol. 66, no. 6, pp. 4672–4683, Jun. 2019.

[25] Y. An, J. Li, F. R. Yu, J. Chen, and V. C. M. Leung, "A novel HTTP anomaly detection framework based on edge intelligence for the Internet of Things (IoT)," *IEEE Wireless Commun.*, vol. 28, no. 2, pp. 159–165, Apr. 2021.

[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] L. Cui et al., "Security and privacy-enhanced federated learning for anomaly detection in IoT infrastructures," *IEEE Trans. Ind. Informat.*, vol. 18, no. 5, pp. 3492–3500, May 2022.

[28] J. Hu, H. Zhang, L. Song, R. Schober, and H. V. Poor, "Cooperative Internet of UAVs: Distributed trajectory design by multi-agent deep reinforcement learning," *IEEE Trans. Commun.*, vol. 68, no. 11, pp. 6807–6821, Nov. 2020.

[29] F. Wu, H. Zhang, J. Wu, Z. Han, H. V. Poor, and L. Song, "UAV-to-device underlay communications: Age of information minimization by multi-agent deep reinforcement learning," *IEEE Trans. Commun.*, vol. 69, no. 7, pp. 4461–4475, Jul. 2021.

[30] Y. Chen, B. Ai, Y. Niu, H. Zhang, and Z. Han, "Energy-constrained computation offloading in space-air-ground integrated networks using distributionally robust optimization," *IEEE Trans. Veh. Technol.*, vol. 70, no. 11, pp. 12113–12125, Nov. 2021.

[31] C. Zhan and Y. Zeng, "Energy-efficient data uploading for cellular-connected UAV systems," *IEEE Trans. Wireless Commun.*, vol. 19, no. 11, pp. 7279–7292, Nov. 2020.

[32] C. H. Liu, Z. Chen, J. Tang, J. Xu, and C. Piao, "Energy-efficient UAV control for effective and fair communication coverage: A deep reinforcement learning approach," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 2059–2070, Sep. 2018.

[33] Q. Ren, O. Abbasi, G. K. Kurt, H. Yanikomeroglu, and J. Chen, "Caching and computation offloading in high altitude platform station (HAPS) assisted intelligent transportation systems," *IEEE Trans. Wireless Commun.*, vol. 21, no. 11, pp. 9010–9024, Nov. 2022.

[34] R. Han, L. Bai, J. Liu, J. Choi, and Y.-C. Liang, "A secure structure for UAV-aided IoT networks: Space-time key," *IEEE Wireless Commun.*, vol. 28, no. 5, pp. 96–101, Oct. 2021.

[35] X. Tang, X. Lan, L. Li, Y. Zhang, and Z. Han, "Incentivizing proof-of-stake blockchain for secured data collection in UAV-assisted IoT: A multi-agent reinforcement learning approach," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 12, pp. 3470–3484, Dec. 2022.

[36] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of Wasserstein GANs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1–11. pp. 5769–5779.

[37] W. Wang, C. Liang, L. Tang, H. Yanikomeroglu, and Q. Chen, "Federated multi-discriminator BiWGAN-GP based collaborative anomaly detection for virtualized network slicing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 11, pp. 6445–6459, Nov. 2023.

[38] L. Xie, K. Lin, S. Wang, F. Wang, and J. Zhou, "Differentially private generative adversarial network," 2018, *arXiv:1802.06739*.

[39] Y. Zeng, J. Xu, and R. Zhang, "Energy minimization for wireless communication with rotary-wing UAV," *IEEE Trans. Wireless Commun.*, vol. 18, no. 4, pp. 2329–2345, Apr. 2019.

[40] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," *Found. Trends Mach. Learn.*, vol. 11, nos. 3–4, pp. 219–354, 2018.

[41] J. Luo, Q. Chen, L. Tang, Z. Zhang, and Y. Li, "Adaptive resource allocation considering power-consumption outage: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 72, no. 6, pp. 8111–8116, Jun. 2023.

[42] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[43] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," 2019, *arXiv:1509.02971*.

[44] "Intel Berkeley Research Lab." Accessed: May 10, 2023. [Online]. Available: http://db.csail.mit.edu/labdata/labdata.html

[45] M. Al-Hawawreh, E. Sitnikova, and N. Aboutorab, "X-IIoTID: A connectivity-and device-agnostic intrusion dataset for Industrial Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 5, pp. 3962–3977, May 2022. [Online]. Available: https://dx.doi.org/10.21227/mpb6-py55