Software: Practice and Experience

WILEY

SPECIAL ISSUE PAPER

Prototype-Based Collaborative Learning in UAV-Assisted Edge Computing Networks

Enze Yu[1,2] | Haipeng Dai[1,2] | Haihan Zhang[1,2] | Zhenzhe Zheng[3] | Jun Zhao[4] | Guihai Chen[1,2]

[1]State Key Laboratory for Novel Software Technology, Nanjing University, Jiangsu, China | [2]School of Computer Science, Nanjing University, Jiangsu, China | [3]Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China | [4]School of Computer Science and Engineering, Nanyang Technological University, Singapore, Singapore

Correspondence: Haipeng Dai (haipengdai@nju.edu.cn)

Received: 30 September 2024 | **Revised:** 2 November 2024 | **Accepted:** 23 November 2024

Funding: This work was supported in part by the National Key R&D Program of China under Grant No. 2023YFB4502400, in part by the National Natural Science Foundation of China under Grant 61872178, 62272223, U22A2031, in part by the Fundamental Research Funds for the Central Universities under Grant No. 2024300349.

Keywords: collaborative learning | prototype | UAV-assisted edge computing networks

ABSTRACT

Context: The rise of artificial intelligence of things (AIoT) has enabled smart cities and industries, and UAV-assisted edge computing networks are an important technology to support the above scenarios. UAV-assisted refers to leveraging UAVs as a dynamic, flexible infrastructure to assist edge network data processing and communication tasks. Multiple UAVs can use their own resources, and collaborate edge servers to train artificial intelligence (Al) models.

Objective: Compared with cloud-based collaborative computing scenarios, UAV-assisted edge collaborative learning can reduce training and inference delays and improve user satisfaction. However, UAV-assisted edge networks scenario brings new challenges in terms of transmission burden and energy consumption.

Method: This paper proposes a prototype-based joint optimization and training software system. The system consists of an optimization module and a training module. The optimization module first models an optimization problem including energy consumption and prototype error. Then it solves the optimization problem by problem transformation and plans the location of each UAV given the objects' position. After UAVs fly to the designated area and complete data collection, UAVs and the edge server train a model according to the proposed prototype-based collaborative training module. Our training module enables multiple UAVs and an edge server to collaboratively train a model by lightweight prototype transmission and prototype aggregation. We also prove the convergence of the proposed collaborative training method.

Results: Results show our method reduces prototype error and energy consumption by at least 12.31% and improves model accuracy by 3.62% with a little communication burden.

Conclusion: Finally, we verify system performance through experiments.

1 | Introduction

The rapid development of artificial intelligence (AI) has enabled various industries, and its combination with the internet of things (AIoT) can support the applications of smart homes [1], smart cities [2], and smart industries [3]. For smart city and smart industry scenarios, unmanned aerial vehicles (UAVs) have emerged as a pivotal technology across various applications such

Abbreviations: AI, artificial intelligence; UAV, unmanned aerial vehicles.

as surveillance [4], delivery [5], and environmental monitoring [6]. Edge computing technology can make up for the limited computing resources of UAVs and UAVs can play a dynamic, flexible infrastructure to assist edge network data processing and communication tasks. Edge servers can also act as the manager of UAVs and collaborate with UAVs to complete complex tasks. UAV-assisted edge computing networks [7] leverage the flexibility of UAVs in conjunction with edge computing resources to provide localized data processing and intelligence, thereby offering a natural platform for collaborative AI model training by continuous knowledge transmission. The core of this paradigm [8] is that data is processed by UAVs instead of being sent to edge servers for processing and it is particularly important in scenarios that need real-time data processing and low-latency inference.

Traditional cloud-based collaborative computing approaches rely on centralized data and tasks in distant cloud data centers. While effective in most cases, this paradigm suffers from significant drawbacks, including heavy transmission burden and substantial energy consumption. UAV-assisted edge collaborative training brings the computation closer to the data source, reduces training and inference delays, and further lowers the energy consumption during collaborative training. However, UAVs-assisted edge collaborative learning needs more rigorous resource management and collaborative training requirements because of the limited resources of UAVs. First, the burden of transmitting knowledge among UAVs in collaborative learning must be lightweight, because a heavy transmission burden will directly increase computing latency and energy consumption. Second, the limited battery of a UAV is needed to fly, hover, compute, and transmit. Therefore, the energy consumption of all the UAVs needs to be optimized. Third, it is necessary to consider how to ensure good model performance in the resource optimization process. The joint optimization problem of resource and training in a resource-constrained environment should be solved.

Current UAV's collaborative learning methods train models by transmitting auxiliary datasets, models, and intermediate results such as logits. Constructing and sharing auxiliary datasets is a common approach in collaborative training. UAVs collect and select partial datasets to build an auxiliary dataset, which is transmitted to the upper-level server to assist in model training [9, 10]. However, data transmission inevitably incurs high costs and user privacy leaks. Transmitting and substituting models can explicitly improve model performance. The authors [11–13] proposed a federated learning (FL) method for UAV networks. Local models are trained locally and sent to the aggregator. The aggregated model is sent back to the UAVs to conduct the next round of training. The transmission burden of these approaches is large because they directly transmit the model, which not only greatly increases the transmission time but also incurs privacy issues. The intermediate results obtained by the model's partial layer operations contain rich knowledge and therefore are used in heterogeneous model collaborative training [14, 15]. However, the logits transmission is not suited to UAV-assisted edge computing networks because its transmission cost is proportional to data volume. Lately, some researchers [16] proposed collaborative learning by prototypes, and its transmission burden can be reduced by 80% compared with other methods. Prototype is the mean feature vector of the data belonging to the same class. As knowledge transmission carriers, prototypes show great potential in UAV-assisted edge collaborative learning.

Current UAV training optimization methods mainly focus on the training process [17, 18] and energy model [19–21] in traditional collaborative methods. On one hand, they only consider the modeling of the training process including gradient updates latency and model transmission latency [17, 18, 22] instead of considering training performance. On the other hand, the optimization problem of training energy consumption and transmission energy consumption are formulated and solved [19, 20, 23–25]. In fact, optimizing the training performance and energy consumption is often contradicted. UAVs tend to reduce their energy consumption making training performances degraded, so a trade-off needs to be ensured between optimizing training performance and energy consumption. However, the above methods lack the joint consideration of training performance optimization and energy consumption optimization. When conducting collaborative training in UAV-assisted edge computing networks, the impact of training performance also needs to be considered. This can ensure that model performance is improved while optimizing energy consumption.

In this paper, unlike the above UAV collaborative learning methods and training optimization methods, we propose a prototype-based joint optimization and training software system, which includes an optimization and a training module. In the optimization stage, we consider the training performance degradation caused by resource management and name it prototype error. We model the optimization problem of minimizing energy consumption and prototype error. We assign the target locations of all UAVs for data collection and training by solving this problem. In the training stage, we use lightweight prototyping as a means of knowledge transfer and collaborate with multiple UAVs and an edge server to train a model. By aggregating prototypes and global prototype synchronization, we alleviate the performance degradation caused by data heterogeneity. This prototype-based joint optimization and training software system can be utilized in UAV-assisted edge computing networks, to continuously improve model performance by alternately running optimization and training modules.

In summary, our contributions are as follows:

1. We propose a prototype-based joint optimization and training software system in UAV-assisted edge computing networks. This system contains an optimization module and a training module. The optimization module decides the data collection positions of UAVs by solving an optimization problem. The training module enables multiple UAVs to perform prototype-based collaborative training with the edge server using the collected data. To the best of our knowledge, we are the first to use prototypes to perform collaborative training between UAVs and the edge server.
2. In the optimization module, the edge server first models UAV energy consumption and UAV collaborative learning. It then formulates the optimization problem to balance prototype error and energy consumption. After problem transformation, the edge server runs a designed algorithm to solve it. Lastly, the edge server issues instructions to the

UAVs based on the optimization results, informing them to fly to the designated location.

3. In the training module, UAVs first obtain the local prototype by training in its collected dataset. Then all local prototypes are sent to the edge server. Lastly, the edge server aggregates local prototypes and sends the aggregated global prototype to UAVs for the next round of training. Besides, UAVs regularly transmit feature extractor to the edge server and receive the aggregated feature extractor. Our approach not only reduces the transmission burden and alleviates data heterogeneity.
4. Through extensive simulations, we validate the superiority of our proposed system. Our optimization module balances prototype error and energy consumption, outperforming other optimization methods by 12.31% at reducing global cost. Our training module can improve UAV model accuracy by at least 3.62% with merely 4% communication burden compared with other methods.

The rest of this paper unfolds as follows: Section 2 presents the related work. Section 3 introduces the overview of prototype-based joint optimization and training software system. Sections 4 and 5 detail the optimization module and training module, respectively. Section 6 conducts the simulation and shows the results. Section 7 discusses the flexibility and scalability of the proposed system. Lastly, Section 8 summarizes this paper.

2 | Related Work

In this section, we present related work from two perspectives, including learning optimization and collaborative learning in UAV-assisted edge networks. These two perspectives correspond to the optimization module and training module, respectively.

2.1 | Learning Optimization in UAV-Assisted Edge Networks

Many current optimizations work model a collaborative learning process, including model calculation and transmission efficiency when collaboratively training a model. Pham et al. designed sustainable FL-based wireless networks, and goal was to maximize UAV transmit power efficiency by jointly optimizing bandwidth allocation, power control, and UAV placement [17]. Luo et al. proposed to decide on multiple control variables to minimize the total training time and energy consumption [23] in federated learning. Tran et al. proposed to balance two trade-offs [24]. The first trade-off is learning time versus user equipment energy consumption. The second trade-off is computation versus communication learning time. The original problem was then split into multiple sub-problems, and the closed-form solution was obtained by grouping users.

Energy is an important optimization aim in UAV-assisted edge networks. Zhan and Zeng [19] studied on minimizing the energy consumption of the UAV during the entire flight cycle, including the mission completion time, the UAV flight trajectory, and the communication Base Station (BS) association to reduce the energy consumption of UAVs. The works of energy minimization in UAVs are surveyed in [20], and energy optimization is necessary for many bandwidth-hungry and energy-limited applications [21]. formulated an energy-efficient trajectory optimization problem to maximize energy efficiency by optimizing the UAV flight trajectory. Du et al. [26] proposed a Lagrangian dual method to maximize energy efficiency in mobile edge computing scenarios. Singh and Maciocco [27] proposed to solve a network energy minimization problem with a cardinality constraint by cell selection. Wu et al. formulated the joint edge aggregation and association problem in multi-cell federated learning and optimized training time and energy consumption through problem reformulation and function substitution [22].

However, current optimization works rarely consider model training performance, they consider the model training process such as transmission and calculation. Besides, they lack consideration of the contradiction between energy consumption and training performance. In this paper, we not only use lightweight prototypes but also define prototype error as the criterion for training performance and optimize it together with energy consumption.

2.2 | Collaborative Learning in UAV-Assisted Edge Networks

As the computing power of UAVs increases, current works study collaborative training among multiple UAVs. After collecting the data, every UAV will perform local training based on the collected data, and send the knowledge such as gradient or model to the edge server. The edge server then aggregates this knowledge and uses the aggregated results for training to eliminate the impact of heterogeneous data collected by different UAVs. The mainstream collaboration is sharing knowledge through data, models, or logits.

The most direct way of collaborative training is to use UAVs to collect data and upload the collected data to the edge server for training. The trained model can be deployed on UAVs for inference. Huang and Fu [9] complete data collection through task collaboration and design the generalized AoI expectation (GAE) function to complete the path design to obtain data for edge servers. Wu et al. proposed to train larger models in the cloud by transmitting processed data from ends [10]. However, data transmission will increase the cycle of model training and inference, and the response time of model operation will be longer, which will reduce user satisfaction. Besides, the solution of directly transmitting data for collaborative training is not feasible in high privacy protection scenarios [28].

The authors [11] designated a server UAV to conduct efficient collaborative learning in a UAV swarm through participant and sample selection according to the communication sensitivity and resource constraints. Zhang and Hanzo [29] utilize a ground center as an aggregation server. After collecting data, UAVs transmit their model weights. By aggregating these weights, the ground center can obtain a new model. The collaborative training method of the transmission model uses FedAvg [30] to aggregate the model. Although collaborative training can be performed, the transmission of the model consumes a lot of resources and may infringe on user privacy [31]. It is not suitable for larger-scale models collaborative training.

Knowledge Distillation [32] is a popular method to improve the performance of heterogeneous models. Luo et al. [14, 33] employed a knowledge distillation method to produce a lightweight model with high accuracy based on the full model trained in the cloud. Gad et al. [15] proposed a knowledge distillation (KD)-based collaborative learning method to reduce communication overhead for the resource-constrained UAV system. As the knowledge is transmitted by knowledge distillation, the transmission burden of logits is proportional to the amount of data, and it is not suitable for collaborative learning in UAV scenarios.

Prototype [34], as the average value of the feature vectors of the same class of data, is the representative feature vector of all the classes. The transmission burden of prototypes can save at least 80% of the burden compared with the above methods [16], so the prototype shows good potential for UAV cooperative training. Current collaborative learning methods in UAV-assisted edge networks have not used prototypes as knowledge-sharing methods to improve model performance. We propose to use a lightweight prototype to conduct UAV cooperation training. Besides, the difference between the proposed method and the current prototype-based works can be summarized as two aspects. **Different application scenarios**: We design an optimization module considering prototype error and energy consumption to support UAV collaborative training. However, other methods lack consideration of resource consumption when using prototypes and also lack the consideration of UAV scenarios. **Different training processes**: Our training process uses prototypes to assist in data collection, whereas other methods assume that the data has already been collected.

2.3 | Motivation

Therefore, we propose a prototype-based collaborative learning method in UAV-assisted edge computing networks. For learning optimization, our method innovatively takes the training process into consideration for optimization and proposes a new optimization target called prototype error. By jointly optimizing prototype error and energy consumption, we fill the gap that current training optimization methods lack the consideration of training performance. For collaborative learning in UAV-assisted edge networks, we are the first to introduce the prototype for collaborative training. Due to its lightweight nature, the prototype-based collaborative training approach shows great potential in UAV-assisted edge networks.

3 | System Overview

In this section, we introduce our prototype-based joint optimization and training software system. The software system can run on multiple UAVs and an edge server. As shown in Figure 1, the entire system includes an optimization module and a training module. First, the edge server obtains the location of the objects o_1, o_2, and o_3 to be detected in the network, prepares to dispatch the UAVs to collect data on these objects, and uses the collected data for training. Second, the edge server obtains the resource information of UAVs, including communication channels, UAV computing power, initial position *sp*, and so forth. Then the edge server runs the optimization module based on this information and sends control information to the UAVs. Lastly, after UAVs receive the control information, they will fly to the target location to collect data. UAVs run the training module and collaborate with the server to train by transmitting prototypes.

3.1 | Optimization Module

The optimization module collects the resource information from UAVs and solves the joint optimization problem defined in Section 4. The entire optimization module needs to be completed step by step.

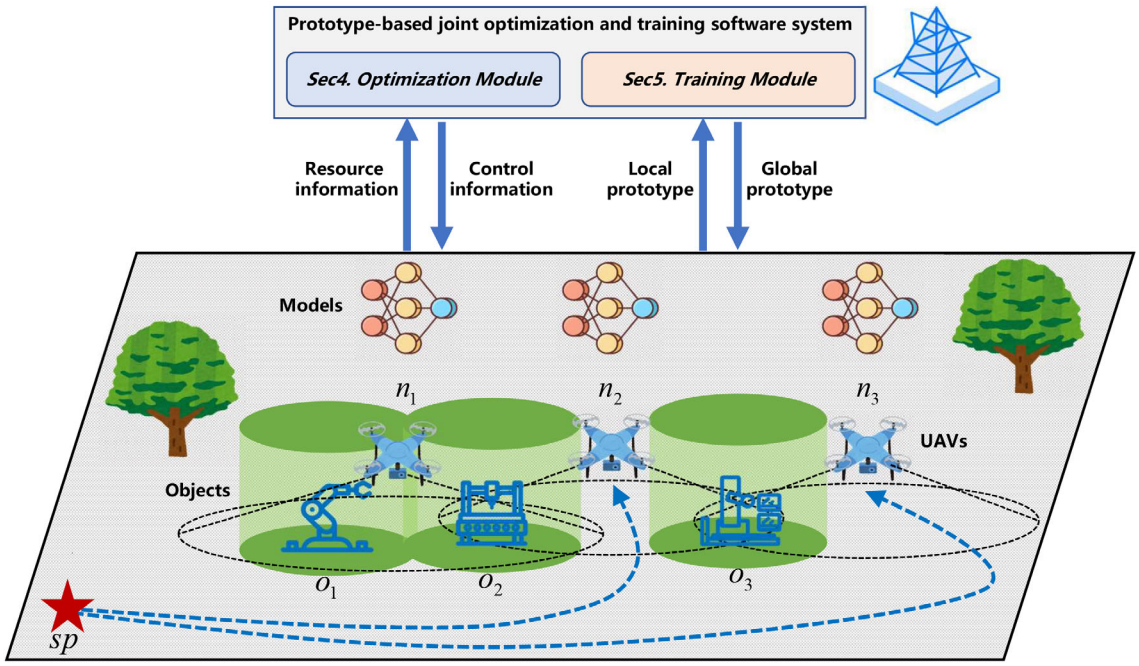

FIGURE 1 | Prototype-based joint optimization and training software system.

1. The edge server needs to know how many UAVs are within its range and establish connections with these UAVs.
2. The edge server model data collection, cooperative training, and consumption energy for all UAVs.
3. An optimization problem with multiple constraints is formulated by the edge server.
4. The edge server solves this optimization problem to produce the results, which guide their data collection next time.

3.2 | Training Module

The training module in our prototype-based joint optimization and training software system is run on both the edge server and UAVs. The entire training module needs to complete the following functions step by step.

1. The edge server produces control information of multiple UAVs and objects based on the optimization module's output results.
2. The control information is sent to the UAVs, which instructs the UAVs which objects should be detected. After receiving the control instructions, the UAVs will fly to the corresponding positions for data collection and training.
3. UAVs send the local prototypes to the edge server for aggregation.
4. The edge server aggregates local prototypes to the global prototype and sends the global prototype to all the UAVs to assist them in the next round of training. The detail of UAV training process is shown in Section 5.

4 | The Optimization of Data Collection and Training

In this section, we introduce the optimization of data collection and training. It considers energy consumption and prototype error and corresponds to the optimization module in Figure 1.

4.1 | Problem Statement

In our scenario, there are N_o objects forming the set of objects $\mathrm{O} = \{o_1, \ldots, o_{|N_o|}\}$. We have N UAVs available forming $N = \{n_1, \ldots, n_{|N|}\}$ for scheduling to collect video data of these objects for training. The entire scenario is modeled as a two-dimensional Euclidean space. An edge server S is deployed for management. Before the commencement of each training round, each UAV flies to its designated position and hovers. During the training phase, the UAVs capture image data of objects within their field of view and collaborate with the edge server to perform training tasks. The main notations used in this paper in Table 1.

4.1.1 | Data Collection Model

Each UAV $n \in \mathrm{N}$, positioned at $pos_n = (x_n, y_n)$, collects data from objects within its field of view, where the maximum effective range is d_N. The set DT_n of objects collected by UAV n can be expressed as

$$DT_n = \left\{ dt_n^{o_i} | o_i \in \mathrm{O}, \mathrm{dist}\left(pos_n pos_{o_i}\right) \leq d_N \right\} \tag{1}$$

TABLE 1 | Main notations.

Symbols	Description
Optimization related	
N_o, O, N, N	The number of objects, the set of objects, the number of UAVs, the set of UAVs
pos_n, d_N	The position of UAV n, the maximum effective range for UAVs
DT_n, $dt_n^{o_i}$	All data collected by UAV n, data of object o_i collected by UAV n
p^{o_i}, p^{O}	The local prototype of object o_i, the global prototype of all the objects
$Err(pos_n)$, $Err(\mathbf{POS})$	The prototype error of position pos_n, the prototype error of all the positions in **POS**
E_n^{cmp}, E_n^{tran}, $E(\mathrm{N}, F, P)$	Training energy consumption of UAV n, transmission energy consumption of UAV n, the sum of energy consumption of UAV n
w_1, w_2	Optimization weight for prototype error, optimization weight for energy consumption
f_{min}, f_{max}, p_{min}, p_{max}	The minimum value of CPU frequency, the maximum value of CPU frequency, the minimum value of transmission power, the maximum value of transmission power
Learning framework related	
D_n, $\mid D_n \mid$, X_i, Y_i	Dataset of UAV n, sample number of the dataset, data, label
w_n, $\mathcal{L}$, $\mathcal{D}$, $\mathcal{F}_n$	Model of UAV n, loss function, distance function, feature extractor of end device n
$\overline{p}_n$, $\overline{p}_g$	Local prototype of UAV n, global prototype
Learning framework related	
t_1, t_2	t_1-th communication round, t_2-th communication round
E, T	Local update step number by a UAV, the global round number

where $pos_{o_i} = (x_{o_i}, y_{o_i})$ denotes the position of the object $o_i \in O$ in the two-dimensional Euclidean space, and the Euclidean distance between UAV n and object o_i is defined as

$$\text{dist}\left(pos_n, pos_{o_i}\right) = \sqrt{\left(x_n - x_{o_i}\right)^2 + \left(y_n - y_{o_i}\right)^2} \tag{2}$$

In this case, $dt_n^{o_i}$ represents the data collected from object o_i by UAV n.

4.1.2 | Collaborative Training Model

Before each training round, the edge server S sends the aggregated feature extractor to all UAVs and synchronizes the feature extractor to each UAV, denoted as $\mathcal{F}(\cdot)$. The local model in UAV n is ω_n. The aggregated feature extractor can alleviate data heterogeneity and it is used to generate prototypes. We assume all UAVs have the equal ability to take photos of the same object, which means the same object will produce the same prototype by the synchronized feature extractor. For any object o_i, its prototype p^{o_i} is as follows:

$$p^{o_i} = \mathcal{F}_n\left(\omega_n, dt^{o_i}\right) \tag{3}$$

For all image data of object o_i collected by all UAVs, the generated global prototype can be represented as

$$p^{O} = \sum_{\cup_{o_i \in O}} p_n^{o_i} \tag{4}$$

Inspired by a definition of approximation error [35], for some position pos_n that UAV n flies, its covered objects are O_n. Therefore, its prototype error is

$$Err\left(pos_n\right) = \left| \sum_{o_i \in O_n} p^{o_i} - p_O \right| \tag{5}$$

We need to set the optimized position for each UAV to minimize the prototype error, which can be written in Equation (6). It is worth noting that different UAVs may detect duplicate objects due to their coverage. These duplicate objects are processed by prototype aggregation on the edge server to eliminate heterogeneity.

$$Err(\mathbf{POS}) = \sum_{n \in \mathbf{N}} Err\left(pos_n\right) \tag{6}$$

4.1.3 | UAV Energy Model

The UAV energy consumption model describes the energy usage during UAV movement, hovering, local training, and prototype uploading.

Before each training round, the total energy of each UAV upon departure is E_{total}. In our scenario, we approximate that the UAV moves at a constant speed v, where the movement power and hovering power are Q_m and Q_h, respectively.

We define the set $F = \{f_n | n \in N\}$, which represents the CPU frequencies for all UAVs, where f_n is the CPU frequency for UAV n. Additionally, we define the set $P = \{p_n | n \in N\}$, representing the transmission power of each UAV, where p_n is the transmission power for UAV n.

UAV Movement Energy Consumption. First, we discuss the movement energy consumption E_n^m of UAV n. The shortest path length l_{shot} from its starting position s to the designated position pos_n can be derived using an obstacle-avoiding shortest path algorithm in the Euclidean space. Thus, the movement time is

$$t_n^m = \frac{l_{\text{shot}}}{v} \tag{7}$$

Considering that the UAV needs to make a round trip, the movement energy consumption is

$$E_n^m = 2Q_m t_n^m = 2Q_m \frac{l_{\text{shot}}}{v} \tag{8}$$

UAV Hovering Energy Consumption. Next, we discuss the hovering energy consumption E_n^m of UAV n. The total time for each training round is t_{round}. Assuming the UAV has sufficient energy to hover until the training concludes and return, the hovering time for UAV n is

$$t_n^h = t_{\text{round}} - t_n^m \tag{9}$$

Thus, the hovering energy consumption of UAV n is

$$E_n^h = Q_h t_n^h = Q_h \left(t_{\text{round}} - \frac{l_{\text{shot}}}{v} \right) \tag{10}$$

The energy consumption model also describes the energy usage during local training and prototype upload.

UAV Training Energy Consumption. The energy consumption for local training by UAV n is defined as

$$E_n^{\text{cmp}} = r \cdot \frac{\beta}{2} f_n^2 \alpha_n \left| D_n \right| \tag{11}$$

The number of local training rounds per global round is r. Effective capacitance coefficient is $\frac{\beta}{2}$. CPU frequency of UAV n is f_n. The number of CPU cycles required to process one sample is α_n. The number of samples of UAV n is $|D_n|$.

Prototype Uploading Energy Consumption. The energy consumption for uploading prototypes from UAV n to edge server S is

$$E_n^{\text{tran}} = \frac{p_n d_{\text{proto}} \left| \mathcal{N}_S \right|}{B_S^{\max} \log\left(1 + \frac{h_S p_n}{N_{nos}}\right)} \tag{12}$$

In the above equation, The transmission power of UAV n is p_n. The size of the prototype data to be uploaded is d_{proto}. The number of UAVs connected to edge server S is $|\mathcal{N}_S|$. The maximum bandwidth of the edge server S is $B_S^{\max}$. The channel gain between UAV n and edge server S is h_S. The background noise is N_{nos}.

The total energy consumption in a global round is given in Equation (13), which is equivalent to $E(\mathbf{POS}, F, P)$.

$$E(N, F, P) = E(\mathbf{POS}, F, P) = \sum_{n \in N} \left(E_n^m + E_n^h + E_n^{\text{cmp}} + E_n^{\text{tran}} \right) \tag{13}$$

4.1.4 | Problem Formulation

We define the optimization problem for UAV-assisted edge collaborative learning to minimize prototype errors and energy consumption. The goal is to find the optimal position set for all UAVs, computing frequencies, and transmission powers, represented as **POS**, F, and P, respectively. The optimization problem is formulated as follows:

$$(\mathbf{P1}) \min_{\mathbf{POS},F,P} \quad w_1 \cdot Err(\mathbf{POS}) + w_2 \cdot E(\mathbf{POS}, F, P) \tag{14a}$$

$$s.t. \quad f_{\min} \le f_n \le f_{\max}, \quad \forall pos_n \in \mathbf{POS} \tag{14b}$$

$$p_{\min} \le p_n \le p_{\max}, \quad \forall pos_n \in \mathbf{POS} \tag{14c}$$

$$x_n, y_n \in \mathbb{R}, \quad \forall pos_n \in \mathbf{POS} \tag{14d}$$

$$|\mathbf{POS}| \le N \tag{14e}$$

The optimization is subject to constraints (14b) and (14c), which ensure that the CPU frequency f_n and transmission power p_n for each UAV $n \in$ N remain within their respective allowable ranges, defined by $f_{\min}$, $f_{\max}$, $p_{\min}$, and $p_{\max}$. Constraint (14d) ensures that the coordinates x_n and y_n of each UAV $n \in$ N are real numbers, reflecting the physical positions of the UAVs in a continuous space. Constraint (14e) represents the need to determine locations for at most N UAVs. The weights w_1 and w_2 are used to balance the importance of minimizing prototype error and energy consumption, respectively. By tuning these weights, we can prioritize different aspects of the UAV-assisted edge collaborative learning process depending on the system requirements. Then we have the following propositions and theorems:

Proposition 1. (**P1**) *problem is NP-hard.*

Proof. Consider a special case of problem (**P1**) with the following parameters: the energy consumption term is ignored, that is, $w_2 = 0$, and the UAV selection is reduced to selecting a set of positions for UAVs to minimize the prototype error Err(N). In this special case, the problem can be reformulated as a disk-covering problem in 2D, where each UAV has a coverage area represented by a disk and we aim to cover all targets (regions or objects) with the minimum number of UAVs. The disk-covering problem is known to be NP-hard in general [36, 37]. Thus, (**P1**) is NP-hard in general. □

4.2 | Solution

4.2.1 | Error-Based Area Partition

In order to better address this problem, we initially partition the entire area into multiple subareas. These subareas are defined based on equivalent error metrics as follows:

Definition 1. **Equivalent Error Subarea**: Given a subarea $\widehat{sub}$, it is defined as an equivalent error subarea if, for any positions pos_{n_1} and pos_{n_2} of UAVs n_1 and n_2, where $pos_{n_1}, pos_{n_2} \in \widehat{sub}$, the condition $Err_{n_1} = Err_{n_2}$ holds true.

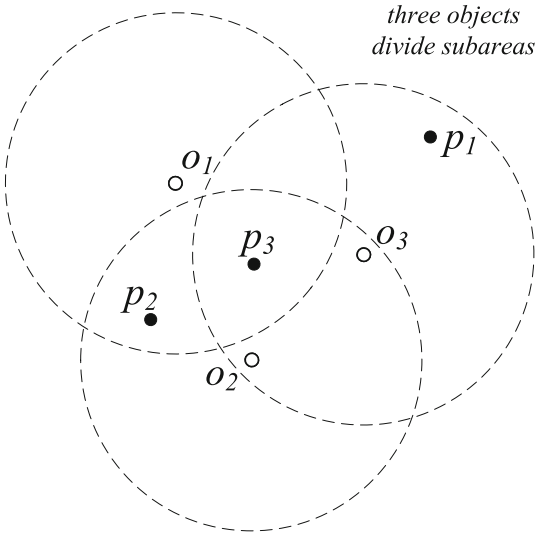

FIGURE 2 | Area partition.

The method for partitioning into Equivalent Error Subareas is as follows: Based on Equation (1), we draw circles with each object o_i at the center and d_N as the radius to divide the entire region into multiple non-overlapping subareas. The set of these areas is denoted as $\widehat{Sub}$. As illustrated in Figure 2, three objects divide the region into eight subareas.

We then propose the following proposition:

Proposition 2. *Each subarea obtained by the area partition method in this section is an Equivalent Error Subarea.*

Proof. In the area partition method, drawing a circle with any object o_i as the center divides the area into two parts. Any position within the circle can collect data from o_i, whereas positions outside cannot, thus each part satisfies Definition 1. Iterating this partitioning and employing mathematical induction leads to the conclusion. □

4.2.2 | Candidate UAV Position Extraction

For each Equivalent Error Subarea $\widehat{sub}$, we identify the position closest to the starting position s as the candidate UAV position. The procedure is as follows:

For each subarea, we divide its boundary into two parts: arcs centered at different circle centers and the intersection points between these arcs. On the one hand, for each arc, we calculate the shortest obstacle-avoiding path from its center to the intersection points on the arc (if any). These intersection points serve as candidate points for that arc.

On the other hand, the intersection points between arcs are also treated as candidate points, for which we similarly compute the shortest paths using existing obstacle-avoidance algorithms.

Subsequently, we compare the path lengths from the starting position s to these candidate points and select the location of the

point with the shortest path as the Candidate UAV Position for that Equivalent Error Subarea $\widehat{sub}$.

By iterating over all $\widehat{sub} \in \widehat{Sub}$, we compile a set of Candidate UAV Positions, denoted as $\mathbf{POS}_{can}$.

4.2.3 | Problem Transformation

We introduce a problem transformation process from (**P1**) to (**P4**). Given any position, (**P1**) is transformed to minimize g by choosing the optimal frequency and power. Referred by [22], this problem is a convex problem and can be solved by convex optimization solvers such as CVX. So we can easily get F^* and P^* if given **POS**. Therefore, (**P1**) is transformed to (**P2**) with $F^*(\mathbf{POS})$ and $P^*(\mathbf{POS})$. We use F^* and P^* to replace $F^*(\mathbf{POS})$ and $P^*(\mathbf{POS})$

$$(\mathbf{P2}) \min_{\mathbf{POS}} \quad g(\mathbf{POS}) = w_1 \cdot Err(\mathbf{POS}) + w_2 \cdot E(\mathbf{POS}, F^*, P^*) \tag{15a}$$

$$s.t. \quad x_n, y_n \in \mathbb{R}, \quad \forall pos_n \in \mathbf{POS} \tag{15b}$$

$$|\mathbf{POS}| \leq N \tag{15c}$$

Definition 2. (**Non-negative, Monotone, Submodular Function**) Given a finite set $\mathcal{X}$, a set function $g : 2^{\mathcal{X}} \to \mathbb{R}$ is termed non-negative, monotone (non-decreasing), and submodular if it satisfies [38]:

1. **Non-negativity**: $g(\emptyset) = 0$ and $g(\mathcal{A}) \geq 0$ for all $\mathcal{A} \subseteq \mathcal{X}$.
2. **Monotonicity**: $g(\mathcal{A}) \leq g(\mathcal{B})$ for all $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{X}$.
3. **Submodularity**: $g(\mathcal{A} \cup \{x\}) - g(\mathcal{A}) \geq g(\mathcal{B} \cup \{x\}) - g(\mathcal{B})$ for all $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{X}, x \notin \mathcal{B}$.

Definition 3. (**Matroid Structure**) A matroid $\mathcal{N} = (\Theta, \mathcal{L})$, where Θ is a finite set and $\mathcal{L} \subseteq 2^{\Theta}$ is a collection of subsets, adheres to:

1. $\emptyset \in \mathcal{L}$.
2. If $L \in \mathcal{L}$ and $L' \subseteq L$, then $L' \in \mathcal{L}$.
3. If $L_1, L_2 \in \mathcal{L}$ and $|L_1| < |L_2|$, there exists $e \in L_2 \backslash L_1$ such that $L_1 \cup \{e\} \in \mathcal{L}$.

Definition 4. (**Uniform Matroid**) A matroid $\mathcal{N} = (\Theta, \mathcal{L})$ is said to be uniform for a given integer k if $\mathcal{L} = \{\mathcal{A} \subseteq \Theta : |\mathcal{A}| \leq k\}$.

We define the uniform matroid $\mathcal{N} = (\mathbf{POS}_{can}, \mathcal{L})$, where $\mathcal{L} = \{\mathcal{A} \subseteq \mathbf{POS}_{can} : |\mathcal{A}| \leq N\}$. Then the characteristics of a set function relevant to our optimization problem is defined as follows:

$$(\mathbf{P3}) \min_{\mathbf{POS}} \quad g(\mathbf{POS}) \tag{16a}$$

$$s.t. \quad \mathcal{L} = \{\mathcal{A} \subseteq \mathbf{POS}_{can} : |\mathcal{A}| \leq N\} \tag{16b}$$

$$\mathbf{POS} \in \mathcal{L} \tag{16c}$$

Proposition 3. (**P3**) *is a problem of the Non-negative, Non-monotone, Supermodular Function Minimization under a Uniform Matroid constraint.*

Proof. First, we prove the non-negative of (**P3**). Obviously, $Err(\mathbf{POS})$ is non-negative because it is the absolute value of prototype error. $\mathcal{E}(\mathbf{POS}, F, P)$ is non-negative because energy is greater than or equal to 0. Therefore, (**P3**) is non-negative.

Second, $Err(\mathbf{POS})$ is monotone decreasing with the number of $Err(\mathbf{POS})$ increases. Adding more elements to $Err(\mathbf{POS})$ will increase the number of detected objects, thus reducing the prototype error, so $Err(\mathbf{POS})$ is monotone decreasing. In contrast, adding more elements to $Err(\mathbf{POS})$ will inevitably consume more energy, so $Err(\mathbf{POS})$ is monotone decreasing. The opposite monotonicity of these two components decides (**P3**) *is non-monotone*.

Third, we discuss the modularity of (**P3**). Having $\forall \mathbf{POS}_1 \subseteq \mathbf{POS}_2$, it means $Err(\mathbf{POS}_1) \geq Err(\mathbf{POS}_2)$. For a new position $pos_n \notin \mathbf{POS}_2$, when pos_n is added into $\mathbf{POS}_1$ and $\mathbf{POS}_2$, three cases need to be considered:

1. The objects detected by pos_n have been covered by $\mathbf{POS}_1$ and $\mathbf{POS}_2$. This means adding pos_n does not affect $Err(\mathbf{POS}_1)$ and $Err(\mathbf{POS}_2)$. Therefore, $Err(\mathbf{POS}_1 \cup pos_n) - Err(\mathbf{POS}_1) = 0 = Err(\mathbf{POS}_2 \cup pos_n) - Err(\mathbf{POS}_2) = 0$.
2. The objects detected by pos_n have been covered by $\mathbf{POS}_2$ but not covered by $\mathbf{POS}_1$. So $Err(\mathbf{POS}_1 \cup pos_n) - Err(\mathbf{POS}_1) = -Err(pos_n)$. For $\mathbf{POS}_2$, $Err(\mathbf{POS}_2 \cup pos_n) - Err(\mathbf{POS}_2) = 0$. Therefore, $Err(\mathbf{POS}_2 \cup pos_n) - Err(\mathbf{POS}_2) > Err(\mathbf{POS}_1 \cup pos_n) - Err(\mathbf{POS}_1)$.
3. The objects detected by pos_n have not been covered by $\mathbf{POS}_2$ and $\mathbf{POS}_1$. In this case, $Err(\mathbf{POS}_2 \cup pos_n) - Err(\mathbf{POS}_2) = Err(\mathbf{POS}_1 \cup pos_n) - Err(\mathbf{POS}_1) = Err(pos_n)$.

Therefore, $Err(\mathbf{POS}_2 \cup pos_n) - Err(\mathbf{POS}_2) \geq Err(\mathbf{POS}_1 \cup pos_n) - Err(\mathbf{POS}_1)$, so $Err(\mathbf{POS})$ *is supermodular*. For $E(\mathbf{POS})$, $E(\mathbf{POS}_1 \cup pos_n) - E(\mathbf{POS}_1) \leq E(\mathbf{POS}_2 \cup pos_n) - E(\mathbf{POS}_2) = E^{cmp}_{pos_n} + E^{tran}_{pos_n}$ Therefore, $E(\mathbf{POS})$ *is supermodular*. In this way, (***P3***) *is supermodular*.

Besides, The constraints in (**P3**) are matroid constraints according to Definition 3. We can conclude that (**P3**) is a problem of the Non-negative, Non-monotone, Supermodular Function Minimization under a Uniform Matroid constraint. □

We invert (**P3**) to make it become a negative non-monotone submodular maximization problem under a uniform matroid constraint. However, (**P3**) is still difficult to solve. We add a term G to ensure $G - g(\mathbf{POS})$ non-negative. Specifically, $G = Err_{max} + E_{max}$. Err_{max} and E_{max} are defined in Equations (17) and (18).

$$Err_{max} = Err(\mathbf{POS}_{can}) \tag{17}$$

$$E_{max} = \max_{pos_n \in \mathbf{POS}_{can}} E^{cmp}_{pos_n} + E^{tran}_{pos_n} \tag{18}$$

Therefore, the above problem is then formulated as

$$(\mathbf{P4}) \max_{\mathbf{POS}} \quad f(\mathbf{POS}) = G - g(\mathbf{POS}) \tag{19a}$$

$$s.t. \quad \mathcal{L} = \{\mathcal{A} \subseteq \mathbf{POS}_{can} : |\mathcal{A}| \leq N\} \tag{19b}$$

$$\mathbf{POS} \in \mathcal{L} \tag{19c}$$

Proposition 4. (**P4**) *is a problem of the (non-negative) Monotone Submodular Function Maximization under a Uniform Matroid constraint (MSFMUM).*

Proof. We can easily know (**P4**) is non-negative because the definitions of Err_{max} and E_{max}. The inversion and the addition of G do not affect monotonicity, so (**P4**) is still non-monotone. The inversion makes (**P4**) become the submodular maximization problem with a uniform matroid constraint. Therefore, we prove this proposition. □

4.2.4 | Submodular-Based Training Optimization

Given that (**P4**) is identified as an MSFMUM problem, we are inspired by the methodologies discussed in [39] to propose Algorithm 1. $P(\mathcal{N})$ is a matroid polytope, the convex-hall of all its independent sets, which is also written as P.

ALGORITHM 1 | Submodular-based training optimization.

Input: Set function $f: 2^{\mathbf{POS}_{can}} \to \mathbb{R}_+$, $n \leftarrow |\mathbf{POS}_{can}|$, $T \leftarrow 1$, $\delta \leftarrow T(\lceil n^5 T \rceil)^{-1}$
Output: Set **POS** under Uniform Matroid Constraint
$t \leftarrow 0, \mathrm{POS}(0) \leftarrow \mathbf{1}_{\emptyset}$
while $t \leq T$ **do**
  for all $e \in \mathbf{POS}_{can}$ **do**
    $\omega_e(t) \leftarrow f\left(\mathbf{POS}(t) \cup \mathbf{1}_e\right) - f(\mathbf{POS}(t)) \geq w$
  end for
  $I(t) \leftarrow argmax\left\{x^* \omega_e(t) | x \in P\right\}$
  for all $e \in \mathbf{POS}_{can}$ **do**
    $\mathbf{POS}_e(t+\delta) \leftarrow \mathbf{POS}_e(t) + \delta I_e(t) \cdot \left(1 - \mathbf{POS}_e(t)\right)$
  end for
  $t \leftarrow t + \delta$
end while
return $\mathbf{POS}(T)$

This algorithm applies a measured continuous greedy strategy, integrating a greedy mechanism with all elements to optimize a submodular function under the constraints of a matroid. Starting from a set (all elements with 0), it incrementally adjusts the probability of elements that contribute the most to the function's value by the weights from the matroid polytope.

The primary goal is to approximate the optimal value of the submodular function as closely as feasible, ensuring computational efficiency. This is achieved by a strategic balance between the exploration of new possible solutions and the exploitation of the most promising candidates found thus far. The solution it produces seeks to be within an acceptable range of the theoretical maximum, utilizing the property of diminishing returns inherent to submodular functions to maximize overall gains from each element added. After getting $\mathbf{POS}(T)$, we can run pipage rounding to obtain a set [39].

Ultimately, we can derive the following theorem:

Theorem 1. *The proposed approach achieves an approximation ratio of* $\frac{1}{e} - o(1)$ *for the problem* (***P1***).

Proof. The strategy to derive an approximation guarantee from (**P1**) to (**P4**) involves a meticulous process, ensuring that the solutions remain optimal or near-optimal at each stage. The transformation from (**P1**) to (**P2**) employs convex optimization techniques which guarantee that, for any configuration of **POS**, the corresponding parameters F and P are optimized. This step confirms that the optimality of (**P2**) is retained for any given set of positions.

In the transition from (**P2**) to (**P3**), by utilizing the Candidate UAV Position Extraction method, the optimal candidate strategies for each subregion are identified. This careful selection ensures that the set of optimal solutions from (**P2**) is included within the decision space for (**P3**), thereby maintaining the optimality of the solutions across these transformations.

The equivalence between (**P3**) and (**P4**) guarantees that any optimal solution applicable to (**P3**) is also applicable and optimal for (**P4**). This consistency is crucial for the integrity of the solution through these stages.

Lastly, the implementation in (**P4**), as informed by the methodologies in [39], is capable of achieving an approximation ratio of $\frac{1}{e} - o(1)$. This final step demonstrates that the entire process from (**P1**) through (**P4**) adheres to the theoretical framework and achieves near-optimal results within the expected bounds.

Thus, we conclude that the original problem (**P1**) achieves an approximation ratio of $\frac{1}{e} - o(1)$, validating the effectiveness and efficiency of the proposed approach across all transformations and problem stages. □

5 | Prototype-Based Collaborative Learning

In this section, we introduce how to collaborate training between multiple UAVs and an edge server through prototype interaction in the UAV-assisted edge computing networks. This section corresponds to the training module in Figure 1.

5.1 | Collaborative Training Method

Given an edge computing scenario with multiple UAVs $\mathrm{N} = \left\{n_1, \ldots, n_{|N|}\right\}$ and one edge server E, every UAV $n \in \mathrm{N}$ has a heterogeneous dataset $D_n = \left\{\left(X_i, Y_i\right)\right\}_{i=1}^{|D_n|}$, where X_i is the data, Y_i is the corresponding label and $|D_n|$ is the number of samples belonging to UAV n. We present how they can improve model performance through the interaction of lightweight prototypes in Algorithm 2.

After all the UAVs have completed data collection, they conduct model training based on the collected data. Every UAV n has its own model ω_n. The UAVs will perform local training with different strategies according to whether the global class prototype $\overline{p}_g$ is stored in the local storage area. If a UAV does not have the global prototype $\overline{p}_g$ from the server, it will minimize the loss function of Equation (20). If UAV n receives and stores the global

ALGORITHM 2 | Collaborative training by prototype interaction.

```
Input: Datasets of each UAV
Output: Models trained on each UAV
Collect data and initialize the local model ω_n
for each round t = 1, 2, …, T do
  for each UAV n = 1, 2, …, N do
    Train locally by Equation (20) or
     Equation (21)
    Obtain local class prototype p̄_n by
     Equation (22)
    Send p̄_n to the edge server
    if t % T_s == 0 then
      Send f_n to the edge server
    end if
  end for
  Compute the global class prototype p̄_g
   by Equation (23)
  if t % T_s == 0 then
    Compute the global feature extractor f̄_g by
     Equation (24)
    Send f̄_g to all the UAVs for the next round
     training
  end if
  Send p̄_g to all the UAVs for the next round
   training
end for
```

prototype $\overline{p}_g$ of the last round, it will minimize the loss function of Equation (21), where $\overline{p}_n$ is the local class prototype of UAV n, $\mathcal{D}(\overline{p}_g, \overline{p}_n)$ measures the distance between global prototype $\overline{p}_g$ and local prototype $\overline{p}_n$ and λ is the control parameter for loss. The design principle of $\mathcal{D}(\overline{p}_g, \overline{p}_n)$ is to alleviate the performance degradation caused by data heterogeneity by bringing the local prototype and the global prototype closer together during local training in every UAV.

$$\mathcal{L}(D_n, \omega_n) = \mathcal{L}_S(\mathcal{F}_n(\omega_n X_i), Y_i) \tag{20}$$

$$\mathcal{L}(D_n, \omega_n) = \mathcal{L}_S(\mathcal{F}_n(\omega_n x), y) + \lambda \cdot \mathcal{D}(\overline{p}_g, \overline{p}_n) \tag{21}$$

After completing a round of local training, every UAV computes its class prototype. Taking the prototype of the jth class as an example, it is computed with Equation (22), where D_n^j represents the jth category data in the UAV n and $|D_n^j|$ represents the number of jth class of data. f_n is the feature extractor part of ω_n. x and y represent a sample and its label in the dataset. The essence of the prototype is the average value of the feature vectors corresponding to different data in the same class. Each UAV computes its own class prototype according to the class it has. If the UAV n detects different objects o_1, o_2, o_3, then its corresponding class prototype should be $\{\overline{p}_n^{o_1}, \overline{p}_n^{o_2}, \overline{p}_n^{o_3}\}$. After completing local training, all UAVs send their class prototypes to the edge server. It should be noted that when the number of loop rounds reaches the synchronization period T_s, all UAVs need to send feature extractors to the edge server, and the edge server will complete the aggregation to further reduce the impact of data heterogeneity.

$$\overline{p}_n^j = \frac{1}{\left|D_n^j\right|} \sum_{(x,y)\in D_n^j} \mathcal{F}_n(x) \tag{22}$$

After the edge server obtains the class prototypes from all the UAVs, these prototypes are aggregated into a global prototype with Equation (23). The global class prototype is the weighted sum of local class prototypes, and the weight of each local class prototype is calculated based on the proportion of the amount of data of this UAV in the global dataset.

$$\overline{p}_g^j = \frac{\left|D_n^j\right|}{\sum_{n\subset \mathrm{N}} \left|D_n^j\right|} \overline{p}_n^j \tag{23}$$

$$\mathcal{F}_g = \frac{\left|D_n^j\right|}{\sum_{n\subset \mathrm{N}} \left|D_n^j\right|} \mathcal{F}_n \tag{24}$$

After the edge server calculates the global prototype, it will send this global prototype to the corresponding UAVs to support their next round of training. When the number of loop rounds reaches the synchronization period T_s, the edge server aggregates the received feature extractors with Equation (24), generates a global feature extractor, and sends it back to all the UAVs. All the drones will continue to train with the global prototype and the global feature extractor it has received.

$$\hat{y}_{new} = \arg\min_j \| \mathcal{F}(\omega_n; x_{new}) - p_g^{(j)} \|_2 \tag{25}$$

In the inference stage, for a new sample x_{new}, the UAV can predict its label $\hat{y}_{new}$ based on the distance between its prototype and every class prototype. In Equation (25), the class prototype that is closest to the prototype of x_{new} is found, and its label will become the prediction label.

5.2 | Convergence Analysis

We conduct a theoretical examination and analyze the convergence bound for our method. We focus on convergence analysis of UAV-assisted edge collaborative learning via transmitting prototypes. We give common assumptions holding for convergence analysis literature [40, 41].

- **Assumption 1.** All the local objective functions of UAVs are L_1-smooth, that is, given two communication round t_1, t_2, $\|\nabla\mathcal{L}_{t_1} - \nabla\mathcal{L}_{t_2}\|_2 \le L_1\|\omega_{t_1} - \omega_{t_2}\|_2$.
- **Assumption 2.** Stochastic gradient is an unbiased estimator of the local gradient for each UAV, that is, $\forall n \in \{1, 2, \ldots, n_{|N|}\}$, $\mathbb{E}_{\xi_n\sim D_n}[g_{n,t_1}] = \nabla\mathcal{L}(\omega_{n,t_1})$.
- **Assumption 3.** The expectation of the stochastic gradient for all UAVs is bounded by G, that is, $\forall n \in \{1, 2, \ldots, n_{|N|}\}$, $\mathbb{E}[\| g_{n,t_1} \|_2] < G$.
- **Assumption 4.** The feature extraction function for all the UAVs is L_2-smooth, that is, given any two communication round t_1, t_2, $\forall n \in \{1, 2, \ldots, n_{|N|}\}$,

$\left\|\mathcal{F}_n(\phi_{n,t_1}) - \mathcal{F}_n(\phi_{n,t_2})\right\| \le L_2\left\|\phi_{n,t_1} - \phi_{n,t_2}\right\|_2$, where $\mathcal{F}_n$ is the embedding function parameterized by ϕ_n.

Lemma 1. *Let Assumptions 1 and 2 hold. From the beginning of the communication round to E local update step, the loss function of any UAV can be bounded as*

$$\mathbb{E}\left[\mathcal{L}_{(t+1)E}\right] \le \mathbb{E}\left[\mathcal{L}_{tE}\right] - \eta\sum_{e=0}^{E-1}\left\|\nabla\mathcal{L}_{tE+e}\right\|_2^2 + \frac{L_1\eta^2 EG^2}{2}$$

Proof. For any end, $\omega_{tE+1} = \omega_{tE} - \eta g_{tE}$ holds, then

$$\begin{aligned}\mathcal{L}_{tE+1} &\overset{(a)}{\le} \mathcal{L}_{tE} + \left\langle\nabla\mathcal{L}_{tE}, \left(\omega_{tE+1} - \omega_{tE}\right)\right\rangle + \frac{L_1}{2}\left\|\omega_{tE+1} - \omega_{tE}\right\|_2^2\\ &= \mathcal{L}_{tE} - \eta\langle\nabla\mathcal{L}_{tE}, g_{tE}\rangle + \frac{L_1}{2}\left\|\eta g_{tE}\right\|_2^2\end{aligned}$$

Taking expectations of both sides of the above equation, then

$$\begin{aligned}\mathbb{E}\left[\mathcal{L}_{tE+1}\right] &\le \mathbb{E}\left[\mathcal{L}_{tE}\right] - \eta\mathbb{E}\left[\langle\nabla\mathcal{L}_{tE}, g_{tE}\rangle\right] + \frac{L_1\eta^2}{2}\mathbb{E}\left[\left\|g_{tE}\right\|_2^2\right]\\ &\overset{(b)}{=} \mathbb{E}\left[\mathcal{L}_{tE}\right] - \eta\left\|\nabla\mathcal{L}_{tE}\right\|_2^2 + \frac{L_1\eta^2}{2}\mathbb{E}\left[\left\|g_{i,tE}\right\|_2^2\right]\\ &\overset{(c)}{\le} \mathbb{E}\left[\mathcal{L}_{tE}\right] - \eta\left\|\nabla\mathcal{L}_{tE}\right\|_2^2 + \frac{L_1\eta^2}{2}G^2\end{aligned}$$

where (a) follows from L_1-smooth bound, (b) follows from Assumption 2, and (c) follows Assumption 3. Then rearranging and summing E steps, we have

$$\mathbb{E}\left[\mathcal{L}_{(t+1)E}\right] \le \mathbb{E}\left[\mathcal{L}_{tE}\right] - \eta\sum_{e=0}^{E-1}\left\|\nabla\mathcal{L}_{tE+e}\right\|_2^2 + \frac{L_1\eta^2 EG^2}{2}$$

□

Lemma 2. *Let Assumptions 3 and 4 hold. After $(t+1)E$ training of UAV, we mark the stage where the global prototype is sent to all UAVs with $(t+1)E + \frac{1}{2}$. The loss function of any UAV can be bounded as*

$$\mathbb{E}\left[\mathcal{L}_{(t+1)E+\frac{1}{2}}\right] \le \mathbb{E}\left[\mathcal{L}_{(t+1)E}\right] + \lambda L_2\eta EG$$

Proof. We add zero terms to $\mathcal{L}_{(t+1)E+\frac{1}{2}}$ as follows:

$$\begin{aligned}\mathcal{L}_{(t+1)E+\frac{1}{2}} &= \mathcal{L}_{(t+1)E} + \mathcal{L}_{(t+1)E+\frac{1}{2}} - \mathcal{L}_{(t+1)E}\\ &\overset{(a)}{=} \mathcal{L}_{(t+1)E} + \lambda \parallel \mathcal{F}_n\left(\phi_{n,(t+1)E}\right)\\ &\quad - \overline{p}_{(t+2)E} \parallel_2 - \lambda \parallel \mathcal{F}_n\left(\phi_{n,(t+1)E}\right) - \overline{p}_{(t+1)E} \parallel_2\\ &\overset{(b)}{\le} \mathcal{L}_{(t+1)E} + \lambda\left\|\overline{p}_{(t+2)E} - \overline{p}_{(t+1)E}\right\|_2\\ &\overset{(c)}{=} \mathcal{L}_{(t+1)E} + \lambda\left\|\sum_{n=1}^{|N|}q_n p_{n,(t+1)E} - \sum_{n=1}^{|N|}q_n p_{n,tE}\right\|_2\\ &\overset{(d)}{=} \mathcal{L}_{(t+1)E} + \lambda \parallel \sum_{n=1}^{|N|}q_n\frac{1}{\mid D_n \mid}\sum_{k=1}^{|D_n|}\left(\mathcal{F}_n\left(\phi_{n,(t+1)E}; x_{n,k}\right)\right.\\ &\quad \left.-\mathcal{F}_n\left(\phi_{n,tE}; x_{n,k}\right)\right) \parallel_2 .\end{aligned}$$

$$\begin{aligned}&\overset{(e)}{\le} \mathcal{L}_{(t+1)E} + \lambda\sum_{n=1}^{|N|}\frac{q_n}{\mid D_n \mid}\sum_{k=1}^{|D_n|} \parallel \mathcal{F}_n\left(\phi_{n,(t+1)E}; x_{n,k}\right)\\ &\quad - \mathcal{F}_n\left(\phi_{n,tE}; x_{n,k}\right) \parallel_2\\ &\overset{(f)}{\le} \mathcal{L}_{(t+1)E} + \lambda L_2\sum_{n=1}^{|N|}q_n\left\|\phi_{n,(t+1)E} - \phi_{n,tE}\right\|_2\\ &\overset{(g)}{\le} \mathcal{L}_{(t+1)E} + \lambda L_2\sum_{n=1}^{|N|}q_n\left\|\omega_{n,(t+1)E} - \omega_{n,tE}\right\|_2\\ &\overset{(h)}{\le} \mathcal{L}_{(t+1)E} + \lambda L_2\eta\sum_{n=1}^{|N|}q_n\sum_{e=0}^{E-1}\left\|g_{n,tE+e}\right\|_2\end{aligned}$$

In the above equations, q_n is the aggregation weight. (a) follows from the definition of the loss function. (b) follows $\|a-b\|_2 - \|a-c\|_2 \le \|b-c\|_2$. (c) and (d) follow the definition of global prototype and local prototype, respectively. (e) and (h) follow from $\left\|\sum a_i\right\|_2 \le \sum\left\|a_i\right\|_2$. (f) follows from L_2-Lipschitz smooth bound. (g) follows from ϕ_n is a subset of w_n.

Take expectations on both sides, then

$$\begin{aligned}\mathbb{E}\left[\mathcal{L}_{(t+1)E+\frac{1}{2}}\right] &\le \mathbb{E}\left[\mathcal{L}_{(t+1)E}\right] + \mathbb{E}\left[\lambda L_2\eta\sum_{n=1}^{|N|}q_n\sum_{e=0}^{E-1}\mathbb{E}[\left\|g_{n,tE+e}\right\|_2\right]\\ &\le \mathbb{E}\left[\mathcal{L}_{(t+1)E}\right] + \lambda L_2\eta\sum_{e=0}^{E-1}\mathbb{E}\left[\parallel g_{n,tE+e} \parallel_2\right]\\ &\overset{(a)}{\le} \mathbb{E}\left[\mathcal{L}_{(t+1)E}\right] + \lambda L_2\eta EG\end{aligned}$$

where (a) follows the Assumption 3. □

Theorem 1. (*Non-convex convergence rate*) *Let Assumptions 1–4 hold and assume $\mathcal{L}^*$ is the local optimal. For any UAV, its convergence rate is bound as follows*:

$$\frac{1}{TE}\sum_{t=0}^{T-1}\sum_{e=0}^{E-1}\left\|\nabla\mathcal{L}_{tE+e}\right\|_2^2 \le \frac{\mathbb{E}\left[\mathcal{L}_0 - \mathcal{L}^*\right]}{\eta ET} + \frac{L_1\eta G^2}{2} + \lambda L_2 G$$

Proof. Substituting Lemma 1 to Lemma 2, we have

$$\mathbb{E}\left[\mathcal{L}_{(t+1)E+\frac{1}{2}}\right] \le \mathbb{E}\left[\mathcal{L}_{tE}\right] - \eta\sum_{e=0}^{E-1}\left\|\nabla\mathcal{L}_{tE+e}\right\|_2^2 + \frac{L_1\eta^2 EG^2}{2} + \lambda L_2\eta EG$$

Then we have

$$\eta\sum_{e=0}^{E-1}\left\|\nabla\mathcal{L}_{tE+e}\right\|_2^2 \le \mathbb{E}\left[\mathcal{L}_{tE} - \mathcal{L}_{(t+1)E+\frac{1}{2}}\right] + \frac{L_1\eta^2 EG^2}{2} + \lambda L_2\eta EG$$

Rearrange and sum t from 0 to $T-1$ (T iteration), then we have the convergence rate as follows:

$$\frac{1}{TE}\sum_{t=0}^{T-1}\sum_{e=0}^{E-1}\left\|\nabla\mathcal{L}_{tE+e}\right\|_2^2 \le \frac{\mathbb{E}\left[\mathcal{L}_0 - \mathcal{L}^*\right]}{\eta TE} + \frac{L_1\eta G^2}{2} + \lambda L_2 G$$

□

In general, the prototype-based UAV-assisted edge collaborative learning method we propose can converge to bounded as the number of global rounds T increases.

6 | Experiments

In this section, we implement the proposed prototype-based joint optimization and training system. Additionally, we will conduct a thorough verification of two specific software modules within this system.

6.1 | Optimization Module

6.1.1 | Experimental Setup

To demonstrate the effectiveness of our optimization module, we simulate a UAV monitoring scenario on a 2D plane, including 10 UAVs and 30 objects. Given its initial position, the optimization module needs to set the target position for each UAV. We set this two-dimensional space to be 100 m × 100 m, which means scale $s_{area} = 100$ m. If there is no special note, 10 UAVs will monitor 30 objects randomly located in this two-dimensional space.

In the parameter setting, we set hovering power Q_h and movement power Q_m as 50 W and 100 W. The movement speed of all UAVs is 5 m/s. The size of the prototype vector corresponding to one object is 100 KB. The local training round is 5 and the global training round is 100. The range of computing frequency is [1, 10] GHz and the transmission power is within [0.1,1] W. Channel gain h_S is set to 10^{-3} and background noise N_{nos} is 10^{-9}. The initial energy that every UAV has is 10^7. Every UAV collects 100 samples after flying to the designed positions.

We choose two representative methods including a grid-based method and a random-based method for comparisons. These two methods are chosen as baselines in many optimization problems, and can also be used to solve non-negative Monotone Submodular Function Maximization under a Uniform Matroid constraint.

1. Grid-based algorithm (Grid): Grid-based search algorithms systematically explore the solution space on a predefined grid to find the optimal solution. It ensures comprehensive coverage. Its computational complexity is high, especially when the dimension increases. Grid search can provide more accurate path planning.
2. Random-based method (Random): Random-based search algorithms search for potential solutions in the solution space through random sampling. They are more flexible and can handle high-dimensional problems, but may miss some solutions. Random search is suitable for quickly exploring a large range of solution space.

Our performance metric is global cost, which is the weighted sum of prototype error and energy consumption. We set w_1 and w_2 as 0.5, which means we balance prototype error and energy consumption.

6.1.2 | Experimental Results

We fix the number of objects N_o to 30 and change the number of UAVs N to measure the global cost of different methods. As shown in Figure 3, the increase in the number of UAVs causes the global cost to show a downward trend. This is because more UAVs will cover more monitored objects, reducing the prototype error. On the other hand, it is because all UAVs obtain the best computing frequency and transmission power, so the energy consumption is controlled. Our method reduces the global cost by 12.31% and 36.11% compared with the grid-based and random-based methods.

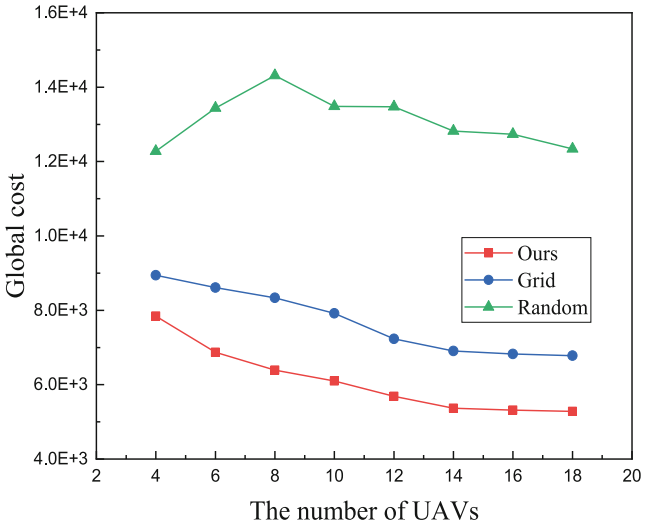

FIGURE 3 | Global cost versus N.

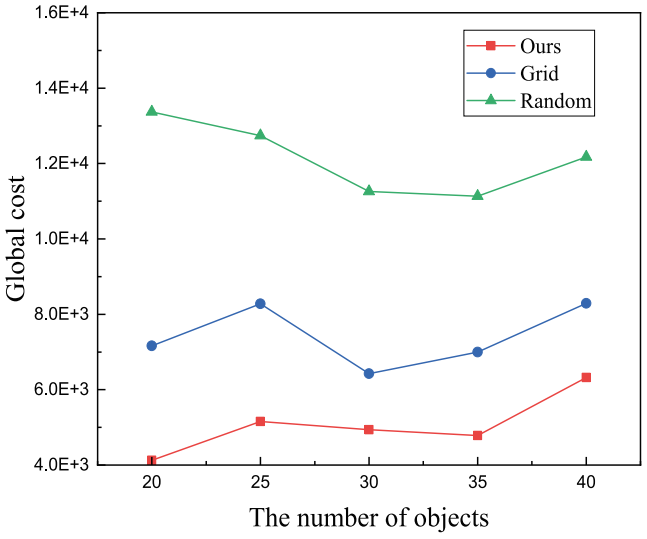

FIGURE 4 | Global cost versus N_o.

We fix the number of UAVs N to 10 and change the number of objects N_o to measure the global cost of different methods. As shown in Figure 4, the increase in the number of objects N_o under the premise of the unchanged number of UAVs causes the prototype error to gradually increase, so the overall global cost shows an increasing trend. When the number of objects N_o is 40, our method reduces the global cost by 23.71% and 48.01% compared with the grid-based and random-based methods.

We change the monitoring range of the UAVs when the number of UAVs is 10 and the number of objects is 30. Different

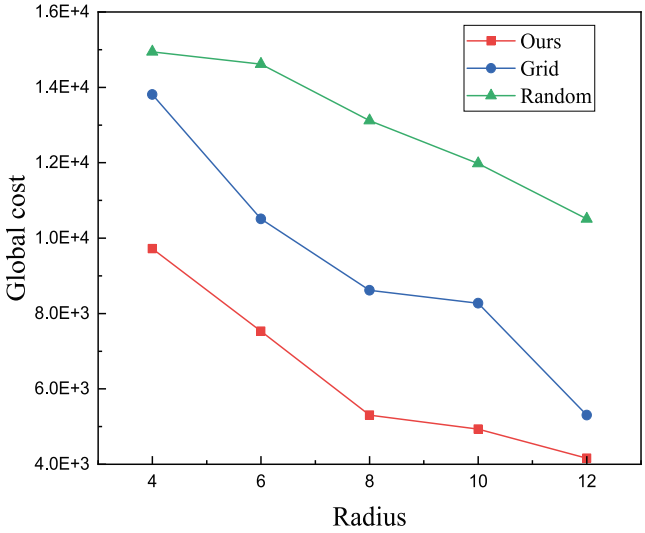

FIGURE 5 | Global cost versus d_N.

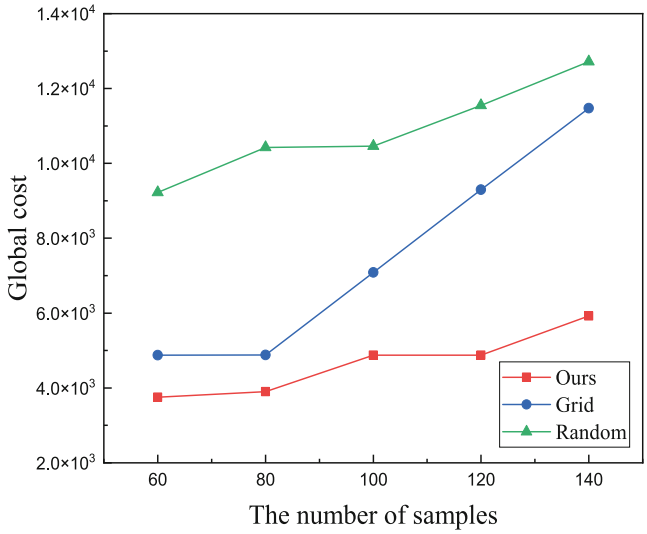

FIGURE 6 | Global cost versus $|D_n|$.

TABLE 2 | Energy consumption and prototype error under different weights.

w_1, w_2	Energy consumption	Prototype error
0.1, 0.9	310.12	5791.17
0.9, 0.1	5882.24	686.96

ranges mean different detection capabilities of UAVs. As shown in Figure 5, all methods reduce the global cost as the monitoring range of the UAVs increases. Our method achieves the lowest global cost and effectively balances prototype error and energy consumption.

We modify different weights and show how our approach optimizes prototype error and energy consumption under different weights. As shown in Table 2, when $w_2/w_1 = 9$, our method will tend to optimize energy consumption instead of prototype error. In contrast, our approach tends to optimize prototype error rather than energy consumption when $w_1/w_2 = 9$. In practical application, we can adjust different weights to alternatively optimize different targets.

In addition, we also evaluate the global cost of every method under different sample numbers D_n, different scene scales s_{area}, and different suspension powers Q_h. Figure 6 shows that the global cost will increase as the number of samples collected by the UAV increases. When the number of samples is 140, the global cost consumed by our method is 5925, which is smaller than other methods. When the scenario scales up, since the number of objects remains the same, all methods need to plan the UAVs to fly a longer distance to collect data. As shown in Figure 7, Our method can still complete the task at the lowest cost under different scenario scales. Lastly, we compare the global cost of different methods under different hovering powers Q_h. As the hovering power increases, all methods will consume more global costs. However, our method reduces the global cost by 35.25% and 36.06% compared with the grid-based method and the random-based method.

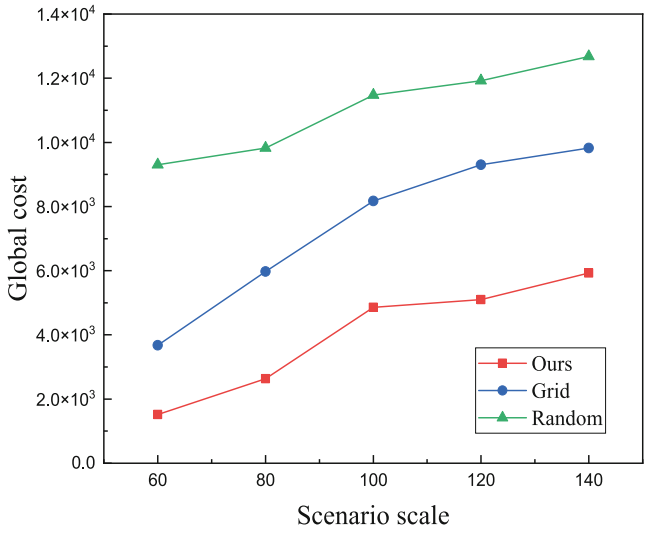

FIGURE 7 | Global cost versus s_{area}.

In summary, our method achieves the lowest global cost when the number of drones, the number of objects, and the monitoring radius are different. By jointly optimizing energy consumption and prototype error, our method balances energy consumption and training performance in Figure 8.

6.2 | Training Module

6.2.1 | Experimental Setup

We conduct experiments on a simulation network scenario consisting of multiple UAVs and an edge server. We set the number of UAVs to 40, and these UAVs can communicate with an edge server. We set up three datasets to make all UAV nodes train different model structures. The datasets we used are introduced as follows:

1. MNIST dataset: MNIST dataset consists of 70,000 grayscale images of handwritten digits. Every image has a 28 × 28

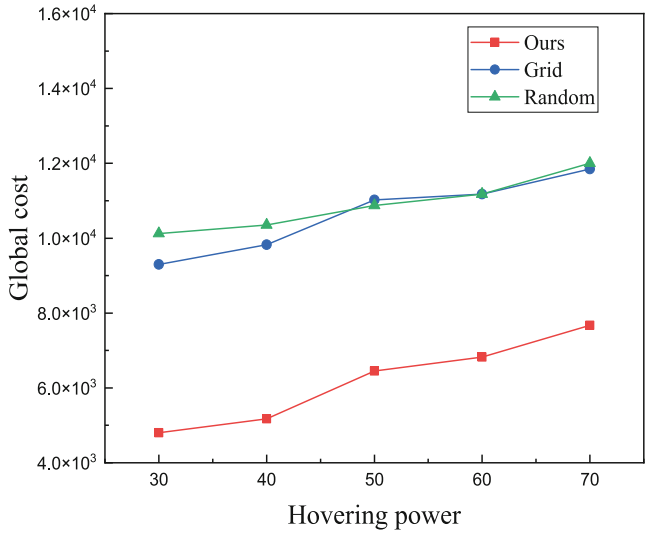

FIGURE 8 | Global cost versus Q_h.

pixel resolution. MNIST dataset is divided into 60,000 training images and 10,000 test images.

2. CIFAR-10 dataset: The CIFAR-10 dataset is widely used in image classification tasks. It contains 60,000 color images with a resolution of 32×32 pixels, spread evenly across 10 different classes. Each class has 6000 images, with 50,000 used for training and 10,000 for testing.

3. CIFAR-100 dataset: The CIFAR-100 dataset is an extension of CIFAR-10, containing 60,000 color images at the same 32×32 resolution. It contains 100 different classes, each containing 600 images. The whole dataset is split into 50,000 training images and 10,000 test images. Each class is a subclass of 20 larger superclasses.

The model structures UAV trained are different in multiple datasets. For MNIST and CIFAR-10 datasets, we use the two-layer CNN structure. For the CIFAR-100 dataset, we use the ResNet-8 structure. We use the Dirichlet distribution parameter *alpha* to control the data heterogeneity. We divide the whole dataset with a ratio of 0.7, which means the ratio of the training set to the test set is 7:3. All the UAVs train models for 100 communication rounds. The batch size is set as 10 and the Adam optimizer with 0.005 initial learning rate. According to the description of Section 2, current collaborative learning methods can be classified into four categories. Our proposed prototype-based training method improves model performance by transmitting prototypes. Therefore, we list three baselines representing model-based, logits-based, and data-based to compare our proposed collaborative learning method. These three methods represent the most commonly used UAV joint training methods and therefore can be chosen as our baselines.

1. Model-based method [11]: The model-based approach enables collaborative training between UAVs and an edge server through two main processes: model aggregation and model release. In model aggregation, the learning contributions from each UAV are combined into a unified model, while model release involves distributing the updated model back to the UAVs. This collaboration enhances the overall performance of the system by allowing UAVs to benefit from shared insights and improve their individual learning processes.

2. Logits-based method [33]: In the logits-based method, the edge server not only performs aggregation but also maintains a larger model. Knowledge distillation is continuously performed through the logits transmission to improve the UAV model's performance.

3. Data-based method [10]: The data-based method needs the auxiliary dataset to train models. This method requires an additional encoder and decoder network on the UAVs. Data is processed through an encoder and decoder to protect the user privacy and then is transmitted to assist model training.

In our experiment, we set the Dirichlet distribution parameter α to 0.1 and 0.3 to simulate data distributions in two different environments. Adjusting the Dirichlet distribution parameter is a commonly used method in federated learning setting [10, 42], and different data distributions are generated to simulate different heterogeneous scenarios.

1. $\alpha = 0.1$: a more heterogeneous data distribution, indicating higher class bias. In this case, certain categories dominate the data points, making it ideal for evaluating the model's robustness and adaptability in handling imbalanced class distributions.

2. $\alpha = 0.3$: the data distribution is more uniform, representing a lower diversity scenario where data points across categories are relatively close together. This setup is suitable for testing the model's performance under balanced class distributions.

These two setups allow for a comprehensive analysis of the model's performance across varying data distributions. We also use two metrics to measure different methods. The first performance metric is the average model precision of all the UAVs after 100 communication rounds. The second metric is the communication burden in a round.

6.2.2 | Experimental Results

Our training method is superior to other comparison methods for accuracy improvement. When $\alpha = 0.1$, our training method obtains higher model performance compared with other methods and the highest accuracy is 98.61%. For the MNIST dataset, in Figure 9, our method makes the model performance remain stable after about 40 rounds of training. This means that higher model performance can be achieved with fewer rounds, thereby greatly reducing the energy consumption of the UAVs. For the CIFAR-10 dataset, in Figure 10, the model trained by our method achieves the highest accuracy and outperforms the Logits-based method and the Data-based method. Our method also achieves better performance than the logits-based method, with the model accuracy improved by 2.4%. As shown in Figure 11, our method also achieves the highest accuracy compared with other methods in the CIFAR-100 dataset.

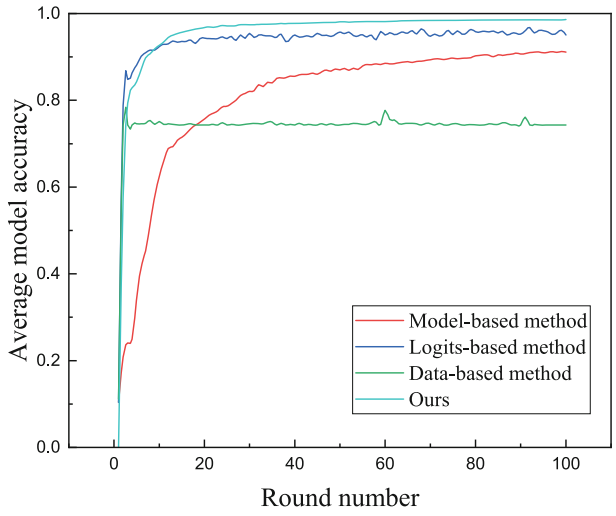

FIGURE 9 | Accuracy (MNIST, $\alpha = 0.1$).

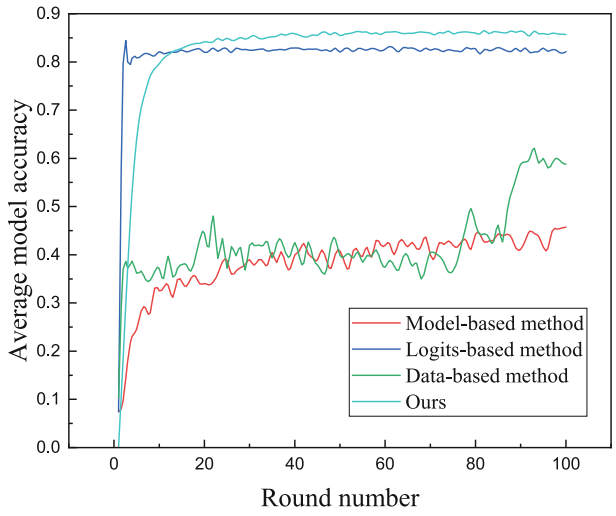

FIGURE 10 | Accuracy (CIFAR-10, $\alpha = 0.1$).

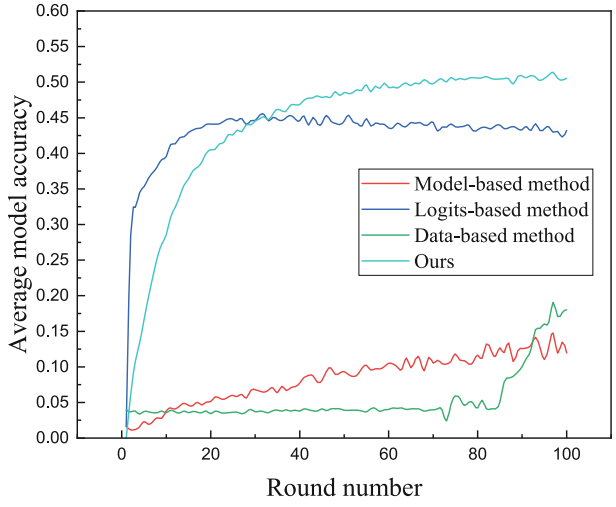

FIGURE 11 | Accuracy (CIFAR-100, $\alpha = 0.1$).

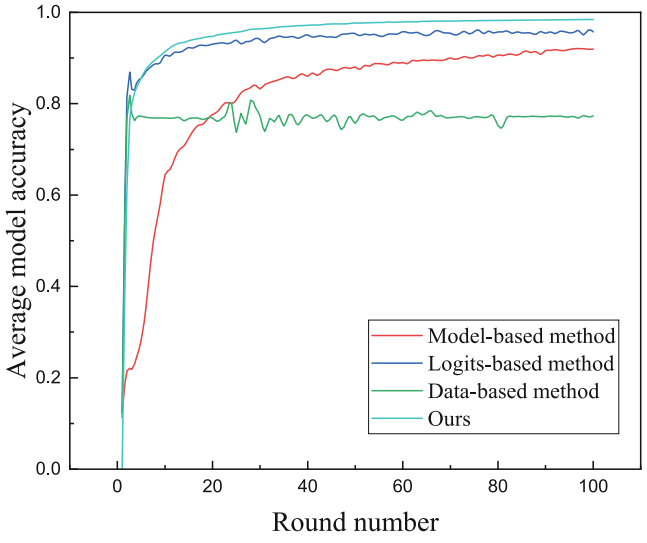

FIGURE 12 | Accuracy (MNIST, $\alpha = 0.3$).

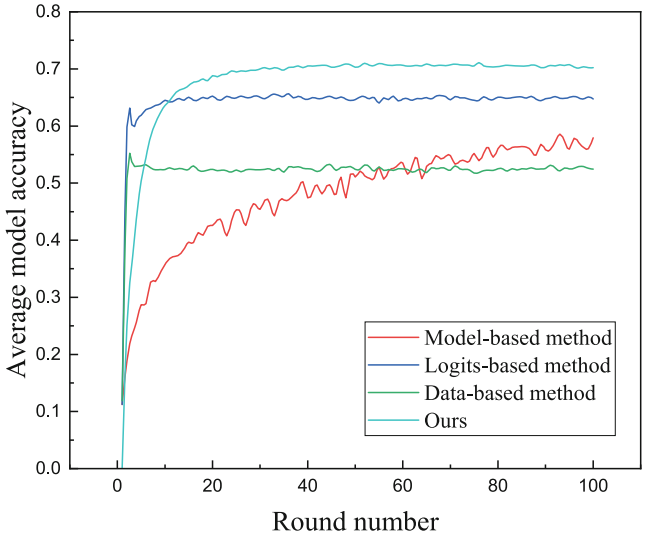

FIGURE 13 | Accuracy (CIFAR-10, $\alpha = 0.3$).

When $\alpha = 0.3$, as shown in Figure 12, in the MNIST dataset, our method also achieves the highest average model accuracy of 98.41%, which outperforms the Model-based method, Logits-based method, and Data-based method by 6.35%, 2.68%, and 20.92%. For the CIFAR-10 dataset, we present results in Figure 13. Our method outperforms the three methods by 13.10%, 5.41%, and 17.25%. Our method is also verified in the CIFAR-100 dataset. As shown in Figure 14, our method outperforms the other three methods by 8.22%, 3.62%, and 22.51%.

Besides, we compute the transmission burden of a communication round in the CIFAR-10 dataset. Experimental results in Table 3 shows our method can complete knowledge sharing between UAVs and the edge server with only 0.11 M information transferred. Comparing other methods, our method saves at least 96% transmission burden. As the number of training rounds increases, our method can greatly improve the overall training speed. This means our method supports efficient training and saves more battery resources for UAVs.

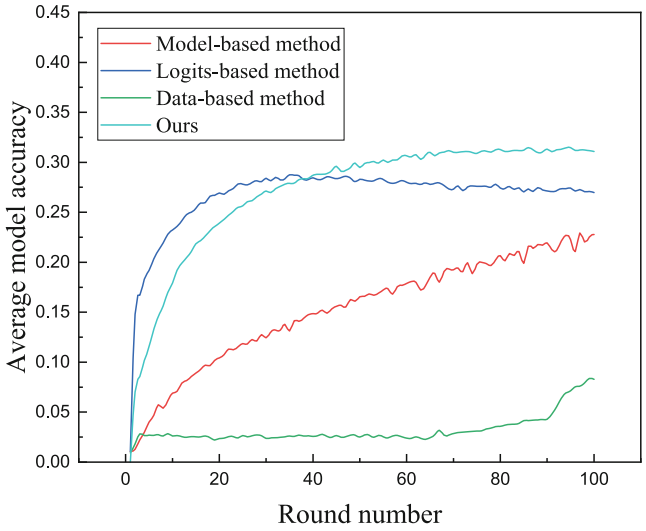

FIGURE 14 | Accuracy (CIFAR-100, $\alpha = 0.3$).

TABLE 3 | Transmission burden of different methods in a communication round.

Methods	Transmission burden in a communication round
Model-based method	3.05M
Logits-based method	32.86M
Data-based method	59.86M
Ours	0.11M

In summary, our approach allows UAVs to gain higher accuracy with a low transmission burden, outperforming all the compared methods. We notice that the logits-based method also achieves good performance, so we can consider combining logits and prototypes to complete collaborative training.

7 | Discussion

We discuss the replaceability of our proposed software system and the scalability of scenarios to further illustrate the advantages of our approach. In addition, we discuss two specific examples including continual learning and interference range communication to demonstrate the scalability of our approach.

Replaceable modules: Our software system consists of an optimization module and a training module, which interact with each other and transmit the information needed by each other to perform collaborative training under the UAV-assisted edge computing network. We can replace any of the modules individually to complete collaborative training in more complex environments. On the one hand, the optimization module can be replaced with a new one, which defines different optimization problems such as considering service satisfaction [43, 44], energy consumption [45], and latency [3]. The new algorithm can replace the current optimization solution to balance the new targets. On the other hand, the training module can be replaced with a new collaborative training method that can use prototypes and logits for collaborative training of heterogeneous models to improve accuracy further.

Scalable scenarios: The collaborative learning method discussed in this paper is based on the parameter server architecture. Our training method can be extended to any architecture. For example, in a decentralized architecture [46], our method allows each UAV to transfer and aggregate prototypes with its neighboring UAVs. Each UAV aggregates the prototypes from its own UAV. The aggregated prototypes can be used not only for model personalized training but also for reasoning based on the distance from the new sample to each class prototype. In a semi-decentralized architecture [47], our method can also be easily scalable. By setting up multiple edge servers for regular aggregation, the performance of heterogeneous models can be improved. In addition, our method is also applicable to other resource-constrained scenarios rather than UAVs. For example, Our method can be easily extended to collaborative learning on resource-limited embedded devices.

Continual learning: In our collaborative training framework involving a single edge server and multiple UAVs, the training module can be designed to support continuous learning [48] with great flexibility. Our module allows the model to adapt to new data and evolving environments without the need for complete retraining. By incorporating mechanisms such as incremental learning and online updates [49], the training module can efficiently integrate new information as it becomes available. Furthermore, the training module can leverage transfer learning techniques, allowing knowledge gained from previous tasks to be applied to new ones, thereby accelerating the learning process. The modular design also facilitates easy integration of new algorithms or techniques as advancements in AI arise. This adaptability ensures that the training module can continuously improve performance while minimizing downtime, making it a vital component of our overall framework for efficient and responsive UAV-assisted edge computing.

Interference range communications: Our optimization problem can be extended to a scenario of interference range communication [50], where the coverage of each edge server is modeled as a circle. In planning the trajectory of the UAVs, it is essential to ensure that all flight paths remain within the range of the edge server to maintain effective communication. We can redefine the optimization problem by modifying the constraints accordingly and identifying different candidate positions for the UAVs. By designing a new algorithm for this problem and replacing optimization modules, we can effectively solve this problem, optimizing both communication efficacy and operational efficiency.

Asynchronous collaborative learning: Our proposed method is suitable for asynchronous UAV collaborative learning. In UAV-assisted edge computing networks, the most notable feature of asynchronous collaborative learning is that the edge server does not wait for all UAVs to complete training before aggregating global prototypes. Therefore, our training module can modify the workflow of the edge server. The edge server can aggregate after obtaining a certain number of prototypes from the UAVs instead of receiving the prototypes sent by all UAVs. For the lagging UAVs,

the edge server still sends the latest global prototype to assist their next training round.

8 | Conclusions

In this paper, we propose a prototype-based joint training and optimization software system for collaborative learning in UAV-assisted edge computing networks. Our system consists of two interacting modules: an optimization module and a training module. We first define the optimization problem of minimizing prototype error and energy consumption. Through problem transformation, we propose an algorithm with approximation guarantees to solve this problem. Our algorithm is integrated with the optimization module, enabling UAVs to obtain the optimal positions for data collection. After data collection, the training module enables multiple UAVs and an edge server to collaboratively train a model by lightweight prototype sharing and global prototype aggregation. We implement these two modules in simulation experiments and verify the superiority of our proposed system from multiple metrics.

Author Contributions

Enze Yu: system design, training module design, algorithm design, writing, experiments. **Haipeng Dai:** model, algorithm design, writing. **Haihan Zhang:** optimization module design, experiments, writing. **Zhenzhe Zheng:** model, algorithm design, writing. **Jun Zhao:** model, system design, writing. **Guihai Chen:** model, system design.

Acknowledgments

This work was supported in part by the National Key R&D Program of China under Grant No. 2023YFB4502400, in part by the National Natural Science Foundation of China under Grant 61872178, 62272223, U22A2031, in part by the Fundamental Research Funds for the Central Universities under Grant No. 2024300349.

Disclosure

The authors have nothing to report.

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

Data sharing is not applicable to this article as no new data were created or analyzed in this study.

References

1. P. Verma, S. K. Sood, H. Kaur, M. Kumar, H. Wu, and S. S. Gill, "Data Driven Stochastic Game Network-Based Smart Home Monitoring System Using IoT-Enabled Edge Computing Environments," *IEEE Transactions on Consumer Electronics* (2024).

2. C. Tang, G. Yan, H. Wu, and C. Zhu, "Computation Offloading and Resource Allocation in Failure-Aware Vehicular Edge Computing," *IEEE Transactions on Consumer Electronics* 70 (2023): 1877–1888.

3. Y. Liu, X. Fang, M. Xiao, et al., "Latency Optimization for Multi-UAV-Assisted Task Offloading in Air-Ground Integrated Millimeter-Wave Networks," *IEEE Transactions on Wireless Communications* 23 (2024): 13359–13376.

4. D. Liu, X. Zhu, W. Bao, B. Fei, and J. Wu, "SMART: Vision-Based Method of Cooperative Surveillance and Tracking by Multiple UAVs in the Urban Environment," *IEEE Transactions on Intelligent Transportation Systems* 23, no. 12 (2022): 24941–24956.

5. Y. Pan, Q. Chen, N. Zhang, Z. Li, T. Zhu, and Q. Han, "Extending Delivery Range and Decelerating Battery Aging of Logistics UAVs Using Public Buses," *IEEE Transactions on Mobile Computing* 22, no. 9 (2022): 5280–5295.

6. W. Wang, H. Dai, C. Dong, et al., "Deployment of Unmanned Aerial Vehicles for Anisotropic Monitoring Tasks," *IEEE Transactions on Mobile Computing* 21, no. 2 (2020): 495–513.

7. Y. K. Tun, Y. M. Park, N. H. Tran, W. Saad, S. R. Pandey, and C. S. Hong, "Energy-Efficient Resource Management in UAV-Assisted Mobile Edge Computing," *IEEE Communications Letters* 25, no. 1 (2020): 249–253.

8. J. Tang, J. Nie, Y. Zhang, Z. Xiong, W. Jiang, and M. Guizani, "Multi-UAV-Assisted Federated Learning for Energy-Aware Distributed Edge Training," *IEEE Transactions on Network and Service Management* 21 (2023): 280–294.

9. X. Huang and X. Fu, "Fresh Data Collection for UAV-Assisted IoT Based on Aerial Collaborative Relay," *IEEE Sensors Journal* 23, no. 8 (2023): 8810–8825.

10. Z. Wu, S. Sun, Y. Wang, et al., "Agglomerative Federated Learning: Empowering Larger Model Training via End-Edge-Cloud Collaboration," in *IEEE INFOCOM 2024-IEEE Conference on Computer Communications* (Vancouver, Canada: IEEE, 2024), 131–140.

11. F. Wu, Y. Qu, T. Wu, et al., "Participant and Sample Selection for Efficient Online Federated Learning in UAV Swarms," *IEEE Internet of Things Journal* 11 (2024): 21202–21214.

12. Y. Qu, H. Dai, Y. Zhuang, et al., "Decentralized Federated Learning for UAV Networks: Architecture, Challenges, and Opportunities," *IEEE Network* 35, no. 6 (2021): 156–162.

13. R. Zhagypar, N. Kouzayha, H. ElSawy, H. Dahrouj, and T. Y. Al-Naffouri, "*UAV-assisted Unbiased Hierarchical Federated Learning: Performance and Convergence Analysis*," 2024, arXiv Preprint, arXiv:2407.07739.

14. H. Luo, T. Chen, X. Li, et al., "KeepEdge: A Knowledge Distillation Empowered Edge Intelligence Framework for Visual Assisted Positioning in UAV Delivery," *IEEE Transactions on Mobile Computing* 22, no. 8 (2022): 4729–4741.

15. G. Gad, A. Farrag, A. Aboulfotouh, K. Bedda, Z. M. Fadlullah, and M. M. Fouda, "Joint Self-Organizing Maps and Knowledge Distillation-Based Communication-Efficient Federated Learning for Resource-Constrained UAV-IoT Systems," *IEEE Internet of Things Journal* 11 (2024): 15504–15522.

16. Y. Tan, G. Long, L. Liu, et al., "Fedproto: Federated Prototype Learning Across Heterogeneous Clients," in *Proceedings of the AAAI Conference on Artificial Intelligence* (Menlo Park, CA: Association for the Advancement of Artificial Intelligence, 2022), 8432–8440.

17. Q. V. Pham, M. Zeng, R. Ruby, T. Huynh-The, and W. J. Hwang, "UAV Communications for Sustainable Federated Learning," *IEEE Transactions on Vehicular Technology* 70, no. 4 (2021): 3944–3948.

18. K. Wei, J. Li, C. Ma, et al., "Low-Latency Federated Learning Over Wireless Channels With Differential Privacy," *IEEE Journal on Selected Areas in Communications* 40, no. 1 (2021): 290–307.

19. C. Zhan and Y. Zeng, "Energy Minimization for Cellular-Connected UAV: From Optimization to Deep Reinforcement Learning," *IEEE Transactions on Wireless Communications* 21, no. 7 (2022): 5541–5555.

20. A. I. Abubakar, I. Ahmad, K. G. Omeke, et al., "A Survey on Energy Optimization Techniques in UAV-Based Cellular Networks: From Conventional to Machine Learning Approaches," *Drones* 7, no. 3 (2023): 214.

21. S. F. Abedin, M. S. Munir, N. H. Tran, Z. Han, and C. S. Hong, "Data Freshness and Energy-Efficient UAV Navigation Optimization: A Deep Reinforcement Learning Approach," *IEEE Transactions on Intelligent Transportation Systems* 22, no. 9 (2020): 5994–6006.

22. T. Wu, Y. Qu, C. Liu, et al., "Joint Edge Aggregation and Association for Cost-Efficient Multi-Cell Federated Learning," in *IEEE INFOCOM 2023—IEEE Conference on Computer Communications* (New York, NY: IEEE, 2023), 1–10.

23. B. Luo, X. Li, S. Wang, J. Huang, and L. Tassiulas, "Cost-Effective Federated Learning Design," in *IEEE INFOCOM 2021—IEEE Conference on Computer Communications* (Vancouver, Canada: IEEE, 2021), 1–10.

24. N. H. Tran, W. Bao, A. Zomaya, M. N. Nguyen, and C. S. Hong, "Federated Learning Over Wireless Networks: Optimization Model Design and Analysis," in *IEEE INFOCOM 2019—IEEE Conference on Computer Communications* (Paris, France: IEEE, 2019), 1387–1395.

25. Y. Liang, H. Tang, H. Wu, Y. Wang, and P. Jiao, "Lyapunov-Guided Offloading Optimization Based on Soft Actor-Critic for ISAC-Aided Internet of Vehicles," *IEEE Transactions on Mobile Computing* 23 (2024): 14708–14721.

26. J. Du, H. Wu, M. Xu, and R. Buyya, "Computation Energy Efficiency Maximization for NOMA-Based and Wireless-Powered Mobile Edge Computing With Backscatter Communication," *IEEE Transactions on Mobile Computing* 23 (2023): 6954–6970.

27. V. Singh and C. Maciocco, "Effective AI/ML Training Using Submodular Cell Selection for Network Energy Saving," in *2024 IEEE International Conference on Communications Workshops (ICC Workshops)* (Denver, CO: IEEE, 2024), 1346–1351.

28. 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 Journal on Selected Areas in Communications* 39, no. 10 (2021): 3144–3159.

29. H. Zhang and L. Hanzo, "Federated Learning Assisted Multi-UAV Networks," *IEEE Transactions on Vehicular Technology* 69 (2020): 14104–14109.

30. B. McMahan, E. Moore, D. Ramage, S. Hampson, and y B A. Arcas, "Communication-Efficient Learning of Deep Networks From Decentralized Data," *Artificial Intelligence and Statistics* 54 (2017): 1273–1282.

31. B. Li, P. Qi, B. Liu, et al., "Trustworthy AI: From Principles to Practices," *ACM Computing Surveys* 55 (2023): 1–46.

32. J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge Distillation: A Survey," *International Journal of Computer Vision* 129, no. 6 (2021): 1789–1819.

33. L. Yao, F. Liu, C. Zhang, Z. Ou, and T. Wu, "*Domain-Invariant Progressive Knowledge Distillation for UAV-Based Object Detection*," 2024, arXiv Preprint, arXiv:2408.11407.

34. J. Snell, K. Swersky, and R. Zemel, "Prototypical Networks for Few-Shot Learning," *Advances in Neural Information Processing Systems* 30 (2017): 4080–4090.

35. Z. Jiang, Y. Xu, H. Xu, Z. Wang, and C. Qian, "Heterogeneity-Aware Federated Learning With Adaptive Client Selection and Gradient Compression," in *IEEE INFOCOM 2023—IEEE Conference on Computer Communications* (New York, NY: IEEE, 2023), 1–10.

36. M. Basappa, R. Acharyya, and G. K. Das, "Unit Disk Cover Problem in 2D," *Journal of Discrete Algorithms* 33 (2015): 193–201.

37. R. Fraser and A. López-Ortiz, "The Within-Strip Discrete Unit Disk Cover Problem," *Theoretical Computer Science* 674 (2017): 99–115.

38. G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An Analysis of Approximations for Maximizing Submodular Set Functions—I," *Mathematical Programming* 14 (1978): 265–294.

39. M. Feldman, J. Naor, and R. Schwartz, "A Unified Continuous Greedy Algorithm for Submodular Maximization," in *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science* (Palm Springs, CA: IEEE, 2011), 570–579.

40. X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "*On the Convergence of FEDAVG on Non-IID Data*," 2019, arXiv Preprint, arXiv:1907.02189.

41. H. Xing, O. Simeone, and S. Bi, "Federated Learning Over Wireless Device-to-Device Networks: Algorithms and Convergence Analysis," *IEEE Journal on Selected Areas in Communications* 39, no. 12 (2021): 3723–3741.

42. H. Baek, W. J. Yun, Y. Kwak, et al., "Joint Superposition Coding and Training for Federated Learning Over Multi-Width Neural Networks," in *IEEE INFOCOM 2022—IEEE Conference on Computer Communications* (London, UK: IEEE, 2022), 1729–1738.

43. J. Tian, D. Wang, H. Zhang, and D. Wu, "Service Satisfaction-Oriented Task Offloading and UAV Scheduling in UAV-Enabled MEC Networks," *IEEE Transactions on Wireless Communications* 22, no. 12 (2023): 8949–8964.

44. C. Tang, H. Wu, and C. Zhu, "Satisfaction Optimization in Failure-Aware Vehicular Edge Computing," in *GLOBECOM 2022—2022 IEEE Global Communications Conference* (Rio de Janeiro, Brazil: IEEE, 2022), 5783–5788.

45. W. Ye, L. Zhao, J. Zhou, S. Xu, and F. Xiao, "Energy-Efficient Flight Scheduling and Trajectory Optimization in UAV-Aided Edge Computing Networks," *IEEE Transactions on Network Science and Engineering* 11 (2024): 4591–4602.

46. P. Wang, H. Yang, G. Han, et al., "Decentralized Navigation With Heterogeneous Federated Reinforcement Learning for UAV-Enabled Mobile Edge Computing," *IEEE Transactions on Mobile Computing* 23 (2024): 13621–13638.

47. M. Kara, A. Laouid, A. Bounceur, M. Hammoudeh, M. Alshaikh, and R. Kebache, "Semi-Decentralized Model for Drone Collaboration on Secure Measurement of Positions," in *Proceedings of the 5th International Conference on Future Networks and Distributed Systems* (Dubai, UAE: Association for Computing Machinery, 2021), 64–69.

48. Z. Gong, O. Hashash, Y. Wang, et al., "UAV-Aided Lifelong Learning for AoI and Energy Optimization in Non-Stationary IoT Networks," *IEEE Internet of Things Journal* 11 (2024): 39206–39224.

49. M. O. Yildirim, E. C. Gok, G. Sokar, D. C. Mocanu, and J. Vanschoren, "Continual Learning With Dynamic Sparse Training: Exploring Algorithms for Effective Model Updates," *Conference on Parsimony and Learning* 234 (2024): 94–107.

50. L. Mishra, S. Benouadah, J. Alghazo, and N. Kaabouch, "Interference Effects on Bandwidth Availability for UAV Communications," in *2024 Integrated Communications, Navigation and Surveillance Conference (ICNS)* (Herndon, VA: IEEE, 2024), 1–6.