SFedXSL: Semi-Synchronous Federated Cross-Sharpness Learning for UAV Swarm

Mingxiong Zhao*, Shihao Zhao*, Chenyuan Feng†, Howard H. Yang‡, and Tony Q. S. Quek§
*Engineering Research Center of Cyberspace, National Pilot School of Software, Yunnan University, China
† Department of Communication, EURECOM, France
‡ Zhejiang University/University of Illinois Urbana-Champaign Institute, Zhejiang University, China
§ Information Systems Technology and Design Pillar, Singapore University of Technology and Design, Singapore
Emails: shihaozhao@mail.ynu.edu.cn, jimmyzmx@gmail.com, Chenyuan.Feng@eurecom.fr,
haoyang@intl.zju.edu.cn, tonyquek@sutd.edu.sg

***Abstract*—Federated learning (FL) emerges as an innovative approach to manage a collection of client UAVs in order to co-train machine-learning models that are readily integrated into an Unmanned Aerial Vehicle (UAV) swarm. However, due to more erratic communication conditions than in terrestrial wireless networks, synchronous aggregation—which is utilized in traditional FL—is no longer feasible in UAV swarm. Additionally, because of the various deployment zones or specifications of UAVs, the data gathered by them are frequently heterogeneous. A significant amount of unlabeled data will be collected by UAV swarm in light of its new flight trajectory and unseen scenarios. To address these issues, we propose a unique Semi-synchronous Federated Cross-Sharpness Learning (SFedXSL) framework to tackle the problems of asynchronous devices and unlabeled data. This framework incorporates unsupervised pre-training and client UAV clustering scheduling. These proposed techniques aim to unlock the full potential of unlabeled and labeled user data, expediting the training process. Simulation results demonstrate that our proposed algorithm surpasses state-of-the-art FL techniques in terms of objection recognition accuracy and service latency.**

***Index Terms*—UAV swarm, semi-supervised learning, semi-synchronous federated learning, unlabeled data, client scheduling.**

I. Introduction

Federated learning (FL) is an innovative decentralized machine-learning paradigm that enables model training across multiple local devices while maintaining data privacy. Under FL, a central server orchestrates the process, merging updates from various devices to develop a globally consistent statistical model. This model effectively mitigates risks associated with centralized data repositories, particularly in contexts where privacy and security are crucial [1]. Within this framework, a wide array of client devices collaborates in model training, with the central server overseeing the coordination and aggregation of updates, thereby prioritizing data confidentiality [2]. This method leverages the collective intelligence of distributed devices, seamlessly integrating with use cases like online training with Unmanned Aerial Vehicle (UAV) swarms [3].

The evolution of edge computing and device-to-device communication technologies has facilitated the extensive deployment of UAV swarm in various applications, including natural disaster surveillance and urban traffic management. These UAVs not only collect data but also possess computational capabilities to train machine-learning models for tasks such as object recognition and trajectory planning. However, the application of FL in UAV swarm presents numerous challenges, primarily due to the unpredictability of aerial transmission links and the diversity of data collected by different UAVs [4]. Unlike terrestrial wireless networks, UAVs often face communication disturbances that compromise training performance. This occurs because the synchronous aggregation of model updates is hampered by connectivity issues and the presence of straggler or disconnected UAVs [5]. Additionally, the heterogeneity of the data, influenced by varying viewpoints and flight paths of the UAVs, complicates the learning process [6]. Furthermore, given the frequent use of UAVs in exploring new or dynamically changing environments, a significant portion of the data they collect may be unlabeled, adding another layer of complexity to the training process.

Despite significant advancements in FL, designing an optimized FL framework that effectively addresses challenges such as data heterogeneity, unlabeled data, and device diversity within UAV swarm remains a complex endeavor. To confront these challenges, we introduce the Semi-synchronous Federated Cross-Sharpness Learning (SFedXSL) framework. This framework aims to enhance data utilization, expedite the training process, and ultimately, improve model accuracy. The key contributions of our work are detailed below:

- We introduce a streamlined semi-supervised federated learning (SSFL) method that effectively utilizes the synergy between abundant unlabeled data and limited labeled data. This approach overcomes the typical challenges of traditional SSFL, where local optima may neglect the potential of unlabeled data due to dependency on dataset similarities and transferability. By reducing the cross-

This work was supported in part by the National Natural Science Foundation of China under Grant 61801418, 62361056, and 62301328, and in part by the Applied Basic Research Foundation of Yunnan Province under Grant 202201AT070203 and 202301AT070422. *Corresponding author: C. Feng.*

sharpness in prediction functions, our method ensures consistent performance across diverse data types and boosts the overall effectiveness of SSFL.

- Addressing the asynchronous communication for a UAV swarm, a theoretical model is proposed to explore the impact of stragglers and dropped connections on training latency under a FL manner. Based on this model, we propose a clustering-based client selection algorithm that not only decreases latency but also rests on a solid theoretical foundation to prove its efficiency.
- The comprehensive simulation demonstrate that the SFedXSL method outperforms traditional FL techniques across several crucial metrics, including accuracy, robustness in object recognition, and service latency. These findings highlight the substantial practical advantages and potential of our proposed framework in optimizing UAV swarm operations in FL scenarios.

The rest of the paper is organized as follows: Section II presents the system model and problem formulation. Section III provides the algorithm description. Section IV demonstrates the effectiveness of our proposed methods. Finally, Section V concludes this paper.

II. System Model and Problem Formulation

As illustrated in Fig. 1, we aim to implement the FL manner in a UAV swarm consisting of $C+1$ devices, denoted as $\mathcal{C} \triangleq \{0, 1, 2, \cdots, C\}$. In this swarm, all client UAVs have the capability of data collection and computing, and one UAV as the leader needs to act as a central server to coordinate the cooperation of the whole swarm. During the flight, each UAV will collect a large amount of unlabeled data and a few labeled data. Let $D_{c,u} = \{z_{c_i}\}_{i=1}^{N_{c,u}}$ and $D_{c,l} = \{(x_{c_j}, y_{c_j})\}_{j=1}^{N_{c,l}}$ denote the unlabeled and labeled datasets on the c-th UAV, respectively, where z_{c_i} represents the i-th unlabeled sample in the input space $\mathcal{X}$, x_{c_j} is the i-th data feature in the input space $\mathcal{X}$, y_{c_j} is its corresponding label in the label space $\mathcal{Y}$, $N_{c,u}$ and $N_{c,l}$ indicate the size of the unlabeled dataset and that of labeled dataset, respectively, and we have $(N_{c,u} \gg N_{c,l})$. Let $D_l = \bigcup_{c=1}^{C} D_{c,l}$ represent the global labeled dataset, encompassing the labeled datasets of all UAVs. In a similar vein, $D_u = \bigcup_{c=1}^{C} D_{c,u}$ denotes the global unlabeled dataset.

A. Traditional FL with Labeled Data

In classical FL, the objective function of local training at the c-th UAV over its own labeled dataset can be given as:

$$\min_{M_c} J(f_{M_c}, D_{c,l}) := \frac{1}{N_{c,l}} \sum\nolimits_{i=1}^{N_{c,l}} \ell(f_{M_c}(x_{c_i}), y_{c_i}), \quad (1)$$

where $f_{M_c} : \mathcal{X} \to \mathcal{Y}$ is an r-layer neural network model parameterized by M_c at the c-th UAV, and $\ell\left(f_{M_c}(x_{c_i}), y_{c_i}\right)$ denotes the loss function of the c-th UAV's local model f_{M_c} over the training sample (x_{c_i}, y_{c_i}). In this context, the KL-divergence or cross-entropy loss functions are widely adopted to evaluate the inconsistency between the model f_{M_c}'s predictions and the actual data labels y_{c_i}, and we adopts the cross-entropy as the local loss function for the client UAVs.

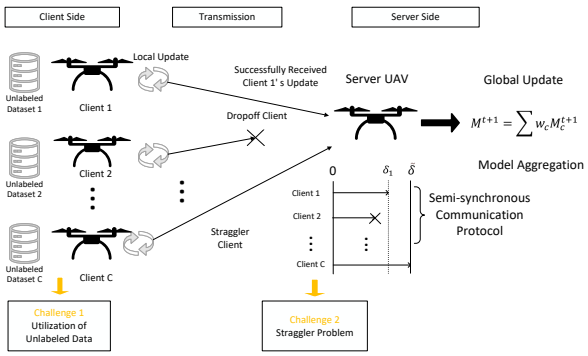


Fig. 1: Overall semi-synchronous FL framework for a UAV swarm and the main challenges.

Through the coordination between the leader UAV and the other UAVs, the overarching objective of FL system is to attain a global model f_M that minimizes the global loss function $J(f_M, D_l)$ across the entire dataset D_l, which can be expressed as follows:

$$\min_{M} J(f_M, D_l) := \sum_{c=1}^{C} \frac{1}{N_{c,l}} \sum_{i=1}^{N_{c,l}} \ell(f_M(x_{c_i}), y_{c_i}), \quad (2)$$

where M denotes the global model parameters.

A conventional optimization approach entails iterative communication between several client UAVs, denoted as U_c, and the server UAV, denoted as U_s, coupled with gradient descent methods, to update the global model. Without loss of generality, we focus on the t-th communication round. At the beginning of the t-th round, the server UAV selects a portion of client UAVs Γ_t from entire UAV swarm for training, let K denote the size of $|\Gamma_t|$. Subsequently, the server UAV distributes the global model f_{M^t} to all selected client UAVs. At each participating client's side, they will perform E_l epochs of local model updates, where $E_l \geq 1$. Let e denote the index of local updating. For $c \in \Gamma_t$, the evolution of local model parameters of communication the t-th round at the c-th UAV is given as:

$$\begin{aligned} M_{c,0}^t &\leftarrow M^t, \\ M_{c,e+1}^t &\leftarrow M_{c,e}^t - \eta \nabla_{M_{c,e}^t} J(f_{M_{c,e}^t}, D_{c,l}), e = 0, \cdots, E_l - 1, \\ M_c^{t+1} &\leftarrow M_{c,E_l}^t, \end{aligned} \quad (3)$$

where M^t denotes the downloaded global model parameters during the t-th round, M_c^{t+1} denotes the c-th UAV's updated local model parameters during the t-th round, $M_{c,e}^t$ denotes the local model parameters at the e-th epoch of local update at the c-th UAV during the t-th round, and η represents the learning rate.

Let $\Gamma_t^{'}$ denote the set of client UAVs whose model updates are successfully received by the server UAV. Due to the unreliable transmission, we have $\Gamma_t^{'} \subseteq \Gamma_t$. Let n denote the number of client UAVs that the server UAV expects to receive successfully, and we have $n \leq K$. In this work, we opt to employ a semi-synchronous communication protocol

to coordinate the FL training process. Specifically, the server UAV will cease waiting and update the global model when either of the following two conditions are met: receiving n client UAV model updates or waiting for $\tilde{T}_t$. In accordance with the FedAvg framework [7], the evolution of global model parameters at the t-th round is given as:

$$M^{t+1} \leftarrow \frac{\sum_{c\in\Gamma_t'} N_{c,l} M_c^{t+1}}{\sum_{c\in\Gamma_t'} N_{c,l}}. \tag{4}$$

B. Unlabeled Data Problem

In real-world scenarios, each UAV often collects a substantial amount of unlabeled data, especially in unseen environments such as agricultural fields, disaster response zones, forest monitoring and smart city [8], [9]. To effectively harness these unlabeled data, the SSFL has emerged as a promising solution [10]. The fundamental concept behind the SSFL is to leverage an initially trained model with respectable classification capabilities to generate surrogate labels for the unlabeled data. The goal of the SSFL is to train a shared model f_M, which can perform well on a global data set. To utilize both labeled and unlabeled data, the global loss function of classical SSFL can be rewritten as follows:

$$\begin{aligned}\min_M J_{\mathrm{SSFL}} =& \min_M (J_l(M, D_l) + J_u(M, D_u)) \\ =& \min_M \sum_{c=1}^{C} \sum_{i=0}^{N_{c,l}} \frac{1}{N_{c,l}} \ell\left(f_M(x_{c_i}), y_{c_i}\right) \\ &+ \sum_{c=1}^{C} \sum_{j=0}^{N_{c,u}} \frac{\mathbb{I}}{N_{c,u}} \ell\left(f_M\left(\mathcal{A}\left(z_{c_j}\right)\right), \widehat{y}_{c_j}\right),\end{aligned} \tag{5}$$

where $J_l(M, D_l)$ is the loss function over the global labeled data under supervised leaning, $J_u(M, D_u)$ denotes the semi-supervised regularization term over the global unlabeled data, $\mathcal{A}(\cdot)$ is an augmentation function that transforms an original input z_{c_j} to an augmented variable, $\widehat{y}_{c_j}$ is the obtained pseudo label, $\mathbb{I}$ is a Boolean variable to select the confident unlabeled data, $\mathbb{I} = 1$ if $\widehat{y}_{c_j} > \tau$, otherwise, $\mathbb{I} = 0$, where τ is a predefined threshold.

Previous research indicates that training exclusively on labeled data can lead to a significant generalization mismatch when applied to unlabeled data, and shows that the introduction of the cross-sharpness term alleviates this problem [6]. As shown in Fig. 2, we plot the loss landscapes of a model trained solely on labeled data and that of a model trained on both labeled and unlabeled data with a cross-sharpness term. It can be observed that the loss landscape becomes smoother and the model achieves smaller loss and higher accuracy. The value of the contour lines for the labeled and unlabeled data in the first row is smaller than that in the second row of Fig. 2, indicating that the training technique based on minimizing cross-sharpness may converge to a better model with a lower training loss. Furthermore, focusing the two sub-figures in the third row, we can find the general trend of the accuracy curves of unlabeled data and labeled data is consistent in semi-supervised learning based on cross-sharpness, while there is a non-negligible mismatch in purely supervised learning scenarios. To guarantee learning consistency between unlabeled and labeled data in a UAV swarm, we introduce a cross-sharpness regularization based FL algorithm described in Section III-A.

C. Straggler Problem

In this scenario, training latency comprises four key components: client UAV model training and uploading latency, global model aggregation time, client UAV selection time, and global model distribution time. Among them, the first term dominates, thus we investigate the details of client UAV model training and uploading latency in this work.

1) Client UAV Updating Latency: Due to device heterogeneity, different UAVs will incur different computation latency and communication latency, which are calculated as:

$$T_c^{\mathrm{cm}} = \frac{M_{\mathrm{size}}}{B \log_2\left(1 + \frac{P_{c,\mathrm{tx}} h_c}{N_0}\right)}, \quad T_c^{\mathrm{cp}} = \frac{\sigma_c}{\varphi_c}, \tag{6}$$

where T_c^{cm} and T_c^{cp} denote the communication latency and computation latency of the c-th UAV, respectively. M_{size} represents the amount of the uploading model parameters in bits, B denotes the bandwidth allocated to each UAV, $P_{\mathrm{c,tx}}$ denotes the transmission power of the c-th UAV, h_c and N_0 denote the channel gain and noise power between the server UAV and client the c-th UAV, σ_c represents the total number of clock cycles used by the c-th UAV for local training, and φ_c represents the CPU frequency of the c-th UAV. Considering the real-world communication environments, UAVs may be disconnected when the communication is unreliable. In this case, the expectation of local training latency is

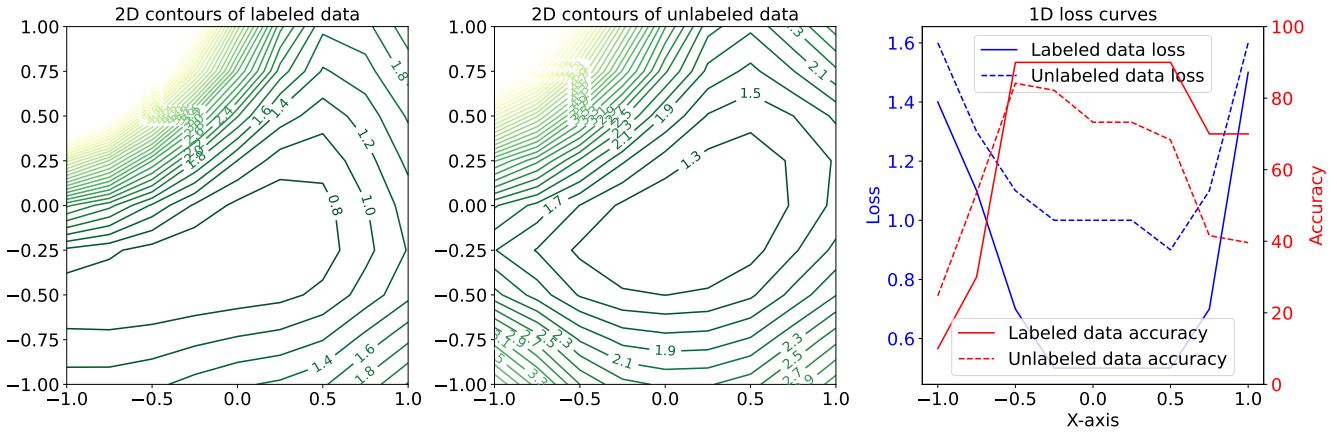
$$T_c = \begin{cases} +\infty, \text{ if the } c\text{-th UAV is dropping out,} \\ T_c^{\mathrm{tr}} = T_c^{\mathrm{cm}} + T_c^{\mathrm{cp}}, \text{ otherwise.} \end{cases} \tag{7}$$

2) Server Waiting Latency: Considering the poor communication conditions and inadequate computation capability of certain users, the server UAV may suffer from the straggler problem. To avoid the server UAV waiting indefinitely for a dropped user, we adopt a semi-synchronous FL manner in this work. Specifically, the server UAV will wait for at most $\tilde{T}_t$ time before generating an aggregated global model, and expect to receive at least n local model updates from K selected client UAVs [11]. The waiting latency for server UAV during the t-th round and its expectation are given as:

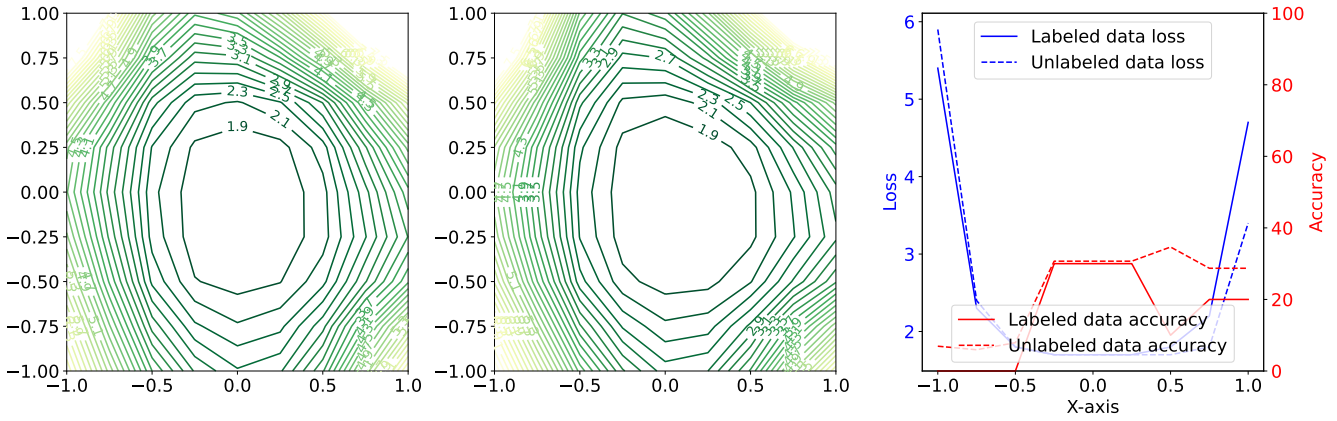
$$\begin{aligned} T_t^{\mathrm{sw}} &= \min\left\{\max_{c_i\in\Gamma_t'}\{T_{c_i}^{\mathrm{tr}}\}, \tilde{T}_t\right\}, \\ \mathbb{E}\left\{T_t^{\mathrm{sw}}\right\} &= (1 - P_{stp}) \max_{c_i\in\Gamma_t'}\{T_{c_i}^{\mathrm{tr}}\} + P_{stp}\tilde{T}_t, \end{aligned} \tag{8}$$

where ρ_c denotes the dropout probability for the c-th UAV at each round, $\tilde{T}_t$ denote the maximum time for the server to wait for client UAV updating during the t-th round[1]. Let

[1]As for the value of $\tilde{T}_t$, a typical approach is to set it to the local update latency of the slowest UAV among all UAVs [11]. In this work, we dynamically set it as the latency of the slowest one in a given participating client UAV set Γ_t.



loss curves of FedXSL (with cross-sharpness)



loss curves with FedAvg (without cross-sharpness)

Fig. 2: Loss landscapes of labeled and unlabeled data were simultaneously generated by training on a single UAV with the SAT6 dataset, using 100 labels per class. (i) The first row and second row show the results after the 60-th epoch of local training with SGD, with and without cross-sharpness, respectively. (ii) The first and the second columns display the 2D loss contours of labeled and unlabeled data, respectively, the third column presents the 1D loss curves.

C_m denote the set of straggler UAVs whose updating latency with reliable transmission is longer than the maximum server waiting time, namely, $C_m = \{c | T_c^{\text{tr}} \geq \tilde{T}_t \text{ and } c \in \mathcal{C}\}$.

In this work, we define the straggler problem as $T_t^{\text{sw}} = \tilde{T}_t$ and it occurs in the following two scenarios: 1) at least $(K-n)$ UAVs are dropped in one training round, denoted by the event A_{drop} ; 2) at least one straggler UAV in C_m is selected, denoted by the event A_{last}. Since events A_{last} and A_{drop} are independent of each other, the probability of the FL server needs to wait for $\tilde{T}_t$ time before model aggregation can be expressed as follows:

$$\begin{aligned} P_{stp} &= P(T_t^{sw} = \tilde{T}_t) = P(A_{\text{drop}} \cup A_{\text{last}}) \\ &= P(A_{\text{drop}}) + P(A_{\text{last}}) - P(A_{\text{drop}})P(A_{\text{last}}). \end{aligned} \tag{9}$$

which is monotonically increasing with respect to $P(A_{\text{last}})$ and $P(A_{\text{drop}})$. Since $P(A_{\text{drop}})$ is determined by the transmission circumstance, and $P(A_{\text{last}})$ is determined by user selection strategy, we aim to reduce the expectation of server waiting latency via optimizing user scheduling to alleviate the straggler problem, which will be described in Section III-B.

III. Algorithm Description and Analysis

A. Cross-Sharpness Learning (XSL) at User Side

In order to bridge the consistency of learning performance between labeled and unlabeled data, a novel method based on cross-sharpness minimization was proposed [6]. This method first find the worst model by performing an adversarial perturbation on the model parameters, and then forces the worst-case model and the original model to agree on the unlabeled data by reducing their prediction difference, termed as cross-sharpness. Thus, the rich amount of unlabeled data can be fully exploited to calibrate the learning direction by minimizing the cross-sharpness regularization. In our work, to efficiently utilize unlabeled samples, each UAV add the cross-sharpness regularization in local loss function as a penalty term, and the worst-case model for each UAV is obtained by maximizing the local loss function on the labeled data. Formally, the cross-

sharpness regularization of the c-th UAV is:

$$\begin{aligned} \min_{M_c} J_{c,\mathrm{XS}} &:= \min_{M_c} \sum_{z_{c_i} \in D_{c,u}} \ell\left(f_{\widetilde{M}_c}(z_{c_i}), f_{M_c}(z_{c_i})\right), \\ \tilde{M}_c &= M_c + \epsilon_c^*, \end{aligned} \tag{10}$$

where $\tilde{M}_c$ denotes the worst-case model with maximum loss function over the c-th UAV's labeled data, ϵ_c^* denotes the optimal model parameter perturbation at the c-th UAV. Following the existing research [12], the optimal model parameter perturbation can be obtained as follows:

$$\begin{aligned} \epsilon_c^* &= \arg\max_{\|\epsilon_c\|_2 \leq \beta} \ell(f_{M_c+\epsilon_c}(x_{c_j}), y_{c_j}) \\ &\approx \beta \frac{\nabla_{M_c} \ell\left(f_{M_c}(x_{c_j}), y_{c_j}\right)}{\|\nabla_{M_c} \ell\left(f_{M_c}(x_{c_j}), y_{c_j}\right)\|_2} \end{aligned} \tag{11}$$

where $\beta > 0$ is a constant to limit the magnitude of perturbation over ϵ_c inside a l_2-sphere.

Cross-sharpness based methods improve the generalization ability by exploiting large amounts of unlabeled data to reduce the prediction difference between unlabeled and labeled data. Under an XSL manner, the local loss function of the c-th UAV can be rewritten by substituting J_u in (5) with $J_{c,\mathrm{XS}}$ in (10), which is as follows:

$$\min_{M_c} J_{c,\mathrm{SSFL}} = \min_{M_c}(J_{c,l} + J_{c,\mathrm{XS}}). \tag{12}$$

Thus, the local update of the c-th UAV at the t-th round is given as:

$$\begin{aligned} M_{c,0}^t &\leftarrow M^t, \\ M_{c,e+1}^t &\leftarrow M_{c,e} - \eta \nabla_{M_{c,e}} J_{c,SSFL}, e = 0, ..., E_l - 1 \\ M_c^{t+1} &\leftarrow M_{c,E_l}^t. \end{aligned} \tag{13}$$

B. Client UAV Cluster Selection (CCS) at Server Side

To address the straggler problem caused by device dropout and heterogeneity described in Section II-C, we propose a utility-based client UAV clustering strategy and Client UAV Cluster Selection (CCS) algorithm, as shown in Algorithm 2.

Algorithm 1 Cross-Sharpness Learning (XSL) Algorithm

Require: Labeled dataset $D_{c,l}$ and unlabeled datasets $D_{c,u}$, model parameters M_c, predefined constant β, Number of local training epoch
Ensure: Local updated model parameters M_c^{t+1}
1: Set local model parameters $M_c^0 \leftarrow M^t$
2: **for** $e = 0$ to $E - 1$ **do**
3: Calculate local labeled loss $J_{c,l}$ according to (1)
4: Calculate optimal model parameter perturbation ϵ_c^* according to (11).
5: Calculate worst-case model $\tilde{M}_c$ and cross-sharpness regularization $J_{c,XS}$ according to (10)
6: Update local model parameters according to (13).
7: **end for**
8: **Return** M_c^{t+1}

Algorithm 2 Client UAV Cluster Selection (CCS) Algorithm

Require: Set of client UAVs $\mathcal{C}$, number of client UAVs selected by the server K, training the t-th round
Ensure: Set of selected client UAVs for the t-th round, Γ_t
1: Calculate the utility of all client UAVs, $U = \{u_1, u_2, \ldots, u_c\}$;
2: **if** $t == 0$ **then**
3: Divide all client UAVs into G clusters in descending order of utility, and obtain the sets of client UAV clusters $\mathcal{G} = \{g_1, \ldots, g_G\}, G = \lceil \frac{C}{K} \rceil$
4: **else**
5: Divide all client UAVs into G clusters in descending order of utility and pad the last group with $K - |g_G|$ client UAVs from the first cluster, and obtain the sets of client UAV clusters $\mathcal{G} = \{g_1, \ldots, g_G\}, G = \lceil \frac{C}{K} \rceil$
6: **end if**
7: Randomly select one client UAV group and assign it to Γ_t
8: **Return** Γ_t

Algorithm 3 Semi-synchronous Federated Cross-Sharpness Learning (SFedXSL) Algorithm

Require: Set of client UAVs $\mathcal{C}$, set of all collected labeled and unlabeled datasets $D_l = \bigcup_{c=1}^{C} D_{c,l}$ and $D_u = \bigcup_{c=1}^{C} D_{c,u}$, number of selected client UAVs for each round K, number of expected received client UAVs' updates for each training round n, maximum number of communication round T, initialized global model parameters M^0
Ensure: Global model parameters M^T
1: **for** $t = 0$ to $T - 1$ **do**
2: Server UAV calls **CCS** Algorithm to select the client UAV set Γ_t;
3: Server UAV sends the current global model parameters M^t to all selected client UAVs in Γ_t
4: Server set the maximum waiting time $\tilde{T}_t$
5: **for** each selected client UAV $c \in \Gamma_t$ in parallel **do**
6: Perform local training based on **XSL** Algorithm
7: Upload the trained local model parameters M_c^t
8: **end for**
9: **while** $\Gamma_t' < n$ OR $T_t < \tilde{T}_t$ **do**
10: **if** Server receive client UAV updates M_c^t **then**
11: $\Gamma_t' = \Gamma_t' \cup c$
12: **end if**
13: **end while**
14: Server updates the global model parameters M^{t+1} according to (4)
15: **end for**
16: **Return** M^T

1) Algorithm Description: As shown in Algorithm 2, the server first computes the utility of each client UAV and sorts it in descending order of utility to obtain the corresponding utility-based user sequence. The server groups all client UAVs into G clusters based on this user sequence, with each client UAV cluster's size equal to the anticipated number of client UAV updates the server receives throughout a training round. For the t-th round, the server will choose a client UAV group at random from the set of client UAV clusters $\mathcal{G} = \{g_1, \ldots, g_G\}$.

In this work, we define the utility function is $u_c = \frac{1}{\mathbb{E}\{T_c^{\text{tr}}\}}$, where u_c represents the utility of client the c-th UAV.

2) Algorithm Analysis: To evaluate the efficacy of our CCS algorithm, we provide the following theorem to analyze the probability that at least one straggler client UAV is chosen at each round, compared with the random selection technique.

Theorem *1:* In semi-synchronous FL scenarios, let $\mathcal{C}$ and $\mathcal{C}_m$ be a set of all client UAVs and a set of straggler client UAVs, respectively, and $|\mathcal{C}| = C, |\mathcal{C}_m| = m$. Let $P_{\text{RS}}(A_{\text{last}})$ and $P_{\text{CCS}}(A_{\text{last}})$ denote the probability that at least one straggler client UAV is selected when the server selects K different client UAVs under a random scheduling strategy for each round and that of our proposed CCS algorithm, respectively, we have

$$P_{\text{RS}}(A_{\text{last}}) = 1 - \frac{\binom{C-m}{K}}{\binom{C}{K}},\ P_{\text{CCS}}(A_{\text{last}}) = \frac{a}{\lceil \frac{C}{K} \rceil} \leq \frac{aK}{C},\ (14)$$
$$P_{\text{CCS}}(A_{\text{last}}) \leq P_{\text{RS}}(A_{\text{last}}), \forall C, K, m \geq 1.$$

where a denotes the number of client UAV clusters that contain at least one straggler client UAV.

Theorem 1 states that our proposed CCS technique could lower the likelihood of at least one user being chosen from the straggler user set, resulting in a shorter server waiting latency in semi-supervised FL circumstances.

C. Overall Algorithm

To address the aforementioned challenges, we propose **S**emi-synchronous **F**ederated Cross-**S**harpness **L**earning (SFedXSL) framework, to incorporate semi-supervised learning and client UAV clustering. The overall design is illustrated in Algorithm 3.

IV. Simulation Results

A. Simulation Setup

In the work, we consider the image classification task for a UAV swarm and adopt two open source datasets, namely SAT6 [13] and FLAME [14] datasets. There are 1 leader UAV and 10 client UAVs to collect data and participate in the training process. We extract ρ percent of the whole dataset as labeled data and the rest as unlabeled data. The number of selected client UAVs at each round is set to 50% of the total user number, i.e. $K = 0.5 * C$. The user CPU frequency ϕ_c follows a Gaussian distribution with a mean of 2.5 GHz and a standard deviation of 0.25 GHz. The bandwidth allocated to each client UAV B is set to 50 MHz. The user transmission power $P_{\text{c,tx}}$ follows a normal distribution $\mathcal{N}(0.1, 0.01)$. The channel gain h_c follows a uniform distribution $U(0.5, 0.8)$. The user dropout probability ρ_c follows $U(0, 0.5)$.

1) Data Split: We extract a certain percent of the whole dataset as labeled data and the rest as unlabeled data, and the object recognition accuracy is calculated over the whole dataset. As for the setting of user data heterogeneity, we utilized the common Dirichlet distribution, in which variable $\epsilon_{q,c}$ denote the ratio of instances of data with label q at client the c-th UAV to the total instance of data with label q, the probability vector follows Dirichlet distribution, i.e. $\epsilon_q = (\epsilon_{q,1}, \epsilon_{q,2}, \ldots, \epsilon_{q,C}) \sim \text{Dir}_C(\alpha)$, where $\text{Dir}_C(\alpha)$ denotes Dirichlet distribution with C users and α denotes the concentration parameter. A smaller α implies a higher level of data heterogeneity, and $\alpha = \infty$ represents the case of homogeneous data. And we set $\alpha = 0.1, \forall q \in \{1, ..., Q\}$ in this work, where Q is the total number of label types.

2) Performance Metrics: The performance metrics used in this paper are learning accuracy and training time. The accuracy is calculated by:

$$\text{Accuracy} = \frac{n_{tp} + n_{tn}}{n_{tp} + n_{tn} + n_{fp} + n_{fn}}, \quad (15)$$

where n_{tp}, n_{tn}, n_{fp} and n_{fn} denote the sample number of true positive, true negative, false negative and false positive, respectively.

3) Baselines: All experiments were implemented on Pytorch. Each result is the average with a standard deviation of 5 simulations. To evaluate the effectiveness of our proposed scheme, we compare our SFedXSL algorithm with the following methods:

- **Independent Training**: Each client UAV independently performs local training with its own labelled data without collaboration. The learning accuracy is the average of all client UAVs and the training delay of each round depends on the slowest one.
- **FedAvg [7]**: As a baseline FL framework, the server randomly selects K different client UAVs at each round.
- **FedProx [15]**: As a variation of FedAvg, the server randomly selects the client UAVs and uses a random function to determine the number of selected users at each round.
- **FedAvg+CCS**: The server adopts our proposed CCS algorithm to select K participating client UAVs at each round.
- **FedAvg+XSL**: The server randomly selects K different client UAVs at each round, and each client UAV performs local training based on XSL scheme.

B. Performance Comparison

First, comparing with FedAvg, FedProx and independent training, our SFedXSL performs the best in in terms of learning accuracy and training time, which verifies the efficacy of our proposed unlabeled data utilization and user scheduling strategy. While the independent training algorithm always obtains the worst learning accuracy, reflecting the necessity of collaboration. Meanwhile, the learning accuracy of all methods increases with the amount of labeled data, which illustrates the importance of labeled data for specific learning tasks.

Second, to assess the efficacy of our proposed user scheduling strategy, we compare FedAvg+CCS with FedAvg and FedProx algorithms, and find that our CCS strategy can obviously achieve faster convergence at the cost of an imperceptible reduction in learning accuracy learning accuracy. This is attributed to the fact that our CCS algorithm can shorten the time spent by users waiting for each other at each round by grouping users with similar local updating time into the same

TABLE I: Accuracy Performance (%) on SAT6 and FLAME Datasets.

Dataset	SAT6		
Percentage of labeled data	5%	10%	Fully Supervised
Independent Training	39.57 ±0.02	45.12 ±0.02	48.47 ±0.02
FedAvg	53.32 ±0.10	69.96 ±0.81	68.06 ±2.16
FedProx	53.29 ±0.05	68.89 ±0.47	65.75 ±4.11
FedAvg+CCS	53.26 ±0.08	69.98 ±0.4	67.9 ±1.99
FedAvg+XSL	86.53 ±0.78	**92.77 ±0.13**	**92.88 ±0.95**
SFedXSL (ours)	**86.81 ±0.18**	92.62 ±0.05	92.17 ±0.15
Dataset	FLAME		
Percentage of labeled data	5%	10%	Fully Supervised
Independent Training	49.15 ±0.02	43.77 ±0.02	58.47 ±0.02
FedAvg	77.38 ±0.12	68.94 ±0.94	81.06 ±1.91
FedProx	77.38 ±0.09	69.81 ±1.11	77.88 ±1.9
FedAvg+CCS	77.34 ±0.08	68.33 ±0.54	80.56 ±1.46
FedAvg+XSL	**81.6 ±0.23**	81.69 ±0.45	**84.03 ±2.64**
SFedXSL (ours)	81.58 ±0.14	**82.69 ±0.22**	83.8 ±2.12

TABLE II: Delay Performance (minutes) on SAT6 and FLAME Datasets.

Dataset	SAT6		
Percentage of labeled data	5%	10%	Fully Supervised
Independent Training	39.07 ±9.07	62.32 ±14.44	590.94 ±138.77
FedAvg	35.47 ±7.99	57.23 ±12.08	518.79 ±115.87
FedProx	36.01 ±2.81	61.95 ±12.95	562.09 ±120.02
FedAvg+CCS	32.29 ±7.36	51.28 ±11.18	457.07 ±107.73
FedAvg+XSL	40.88 ±3.54	67.81 ±13.3	612.11 ±127.77
SFedXSL (ours)	37.70 ±4.54	60.48 ±14.0	539.30 ±137.48
Dataset	FLAME		
Percentage of labeled data	5%	10%	Fully Supervised
Independent Training	171.64 ± 40.51	316.12 ±25.99	3853.99 ±492.47
FedAvg	154.75 ±37.34	281.92 ±19.8	2906.16 ±384.64
FedProx	192.99 ±20.59	328.36 ±15.23	3338.81 ±583.62
FedAvg+CCS	135.74 ±32.92	243.51 ±22.63	2707.19 ±376.96
FedAvg+XSL	201.48 ±9.54	336.70 ±22.81	3414.55 ±903.87
SFedXSL (ours)	192.76 ±23.66	298.70 ±40.66	3120.49 ±591.51

cluster.By comparing FedAvg+XSL with SFedXSL, it is found that although CCS leads to a slightly lower mean of training accuracy, it also has a smaller variance. This shows that CCS can significantly reduce the delay while stabilizing the learning accuracy

Third, to assess the efficacy of our proposed local training strategy, we compare FedAvg+XSL with FedAvg, and find that our XSL strategy could significantly improve the learning accuracy. This is attributed to the fact that our XSL algorithm is optimized based on cross-sharpness between labeled and unlabeled data, which enhances the generalization ability of the model while utilizing unlabeled data. However, the XSL algorithm will lead to longer training delay due to the increase of the amount of training data. Fortunately, combing the proposed XSL and CCS strategy, our proposed SFedXSL algorithm can significantly enhance the learning accuracy and reduce the training time.

V. Conclusion

Though FL emerges as a promising solution to coordinate the UAV swarm, a practical implementation is hindered by challenges such as heterogeneous devices, along with the presence of unlabeled data. For the effective collaboration in a UAV swarm, we proposed the SFedXSL framework to integrate client UAV scheduling and semi-supervised learning optimization. Simulation results unequivocally demonstrate that SFedXSL surpasses traditional FL methods in terms of learning accuracy and training efficiency, which underscores its potential as robust solution to effectively handle unlabeled data and asynchronous devices in unstable transmission environments.

References

[1] Z. Zhao, C. Feng, H. H. Yang, and X. Luo, "Federated-Learning-Enabled Intelligent Fog Radio Access Networks: Fundamental Theory, Key Techniques, and Future Trends," *IEEE Trans. Wireless. Commun.*, vol. 27, no. 2, pp. 22–28, 2020.

[2] C. Feng, H. H. Yang, D. Hu, Z. Zhao, T. Q. S. Quek, and G. Min, "Mobility-Aware Cluster Federated Learning in Hierarchical Wireless Networks," *IEEE Trans. Wireless. Commun.*, vol. 21, no. 10, pp. 8441–8458, 2022.

[3] Z. Cui, T. Yang, X. Wu, H. Feng, and B. Hu, "The Data Value based Asynchronous Federated Learning for UAV Swarm under Unstable Communication Scenarios," *IEEE Trans. Mob. Comput.*, pp. 1–15, 2023.

[4] Z. Wang, Z. Zhang, Y. Tian, Q. Yang, H. Shan, W. Wang, and T. Q. S. Quek, "Asynchronous Federated Learning Over Wireless Communication Networks," *IEEE Trans. Wireless. Commun.*, vol. 21, no. 9, pp. 6961–6978, 2022.

[5] Y. Zhang, D. Liu, M. Duan, L. Li, X. Chen, A. Ren, Y. Tan, and C. Wang, "FedMDS: An Efficient Model Discrepancy-Aware Semi-Asynchronous Clustered Federated Learning Framework," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 3, pp. 1007–1019, 2023.

[6] Z. Huang, L. Shen, J. Yu, B. Han, and T. Liu, "FlatMatch: Bridging Labeled Data and Unlabeled Data with Cross-Sharpness for Semi-Supervised Learning," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 36, Vancouver, Canada, 2024.

[7] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-Efficient Learning of Deep Networks From Decentralized Data," in *Proc. Int. Conf. Artif. Intell. Stat. (AISTATS)*. Fort Lauderdale, Florida, USA: PMLR, 2017, pp. 1273–1282.

[8] W. P. Amorim, E. C. Tetila, H. Pistori, and J. P. Papa, "Semi-supervised learning with convolutional neural networks for UAV images automatic recognition," *Comput. Electron. Agric.*, vol. 164, p. 104932, 2019.

[9] Z. Wei, M. Zhu, N. Zhang, L. Wang, Y. Zou, Z. Meng, H. Wu, and Z. Feng, "UAV-assisted data collection for internet of things: A survey," *IEEE Internet Things J.*, vol. 9, no. 17, pp. 15 460–15 483, 2022.

[10] K. Sohn, D. Berthelot, N. Carlini, Z. Zhang, H. Zhang, C. A. Raffel, E. D. Cubuk, A. Kurakin, and C.-L. Li, "Fixmatch: Simplifying semi-supervised learning with consistency and confidence," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 33, New Orleans, USA, 2020, pp. 596–608.

[11] D. Stripelis, P. M. Thompson, and J. L. Ambite, "Semi-synchronous Federated Learning for Energy-Efficient Training and Accelerated Convergence in Cross-Silo Settings," *ACM Trans. Intell. Syst. Technol.*, vol. 13, no. 5, pp. 1–29, 2022.

[12] P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur, "Sharpness-aware Minimization for Efficiently Improving Generalization," in *Int. Conf. Learn. Represent. (ICLR)*, Vienna, Austria, 2021.

[13] S. Basu, S. Ganguly, S. Mukhopadhyay, R. DiBiano, M. Karki, and R. Nemani, "Deepsat: A Learning Framework for Satellite Imagery," in *Proc. 23rd SIGSPATIAL Int. Conf. Adv. Geogr. Inf. Syst.*, 2015, pp. 1–10.

[14] A. Shamsoshoara, F. Afghah, A. Razi, L. Zheng, P. Z. Fulé, and E. Blasch, "Aerial imagery pile burn detection using deep learning: The FLAME dataset," *Comput. Netw.*, vol. 193, p. 108001, 2021.

[15] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated Optimization in Heterogeneous Networks," in *Proc. Mach. Learn. Syst. (MLSys)*, vol. 2, Santa Clara, USA, 2020, pp. 429–450.