FedOrbit: Energy Efficient Federated Learning for Orbital Edge Computing Using Block Minifloat Arithmetic

Mohammad Reza Jabbarpour, Bahman Javadi, *Senior Member, IEEE*, Philip H.W. Leong, Rodrigo N. Calheiros, and David Boland

***Abstract*—Low Earth Orbit (LEO) satellite constellations have diverse applications, including earth observation, communication services, navigation, and positioning. These constellations have evolved into a valuable data source; however, their use in a ground station (GS) for analysis via machine learning algorithms presents challenges due to constraints on power consumption, communication bandwidth, and onboard computing capabilities. While the combination of Federated Learning (FL) and Orbital Edge Computing has been employed to address these challenges, its heavy reliance on the GS for model aggregation and edge resource limitations remains a research challenge. This article presents FedOrbit, a novel energy-efficient and decentralised FL method to optimise communication with the GS and reduce power consumption. FedOrbit utilises reinforcement learning for cluster formation, satellite visiting patterns for master satellite selection, and block minifloat arithmetic for power reduction. Extensive performance evaluation under Walker Delta-based LEO constellation configurations and different datasets reveals that FedOrbit can maintain high accuracy while significantly reduce communication demand, power consumption and training time in comparison to state-of-the-art FL approaches. The proposed technique can also reduce the training time by 5× compared with the centralised FL approaches. In addition, the utilisation of block minifloat representation as low-precision arithmetic enhanced the energy consumption by 3.5× compared with the single-precision (FP32) format.**

***Index Terms*—Block minifloat, energy consumption, federated learning, low-earth orbit, orbital edge computing.**

I. Introduction

LOW Earth Orbit (LEO) satellites have become increasingly popular in recent years due to advancements in technology and their potential for various applications. LEO deployments are characterised by a large network of satellites used in applications such as internet connectivity in underserved areas [1], Earth observation [2], and weather monitoring [3]. LEO satellites generate large amounts of data, which has driven the utilisation of artificial intelligence (AI) and machine learning (ML) in centralised Ground Stations (GS) where the data is stored. The emerging area of *Orbital Edge Computing* (OEC) [3] aims to explore the processing capacity of LEO satellites to reduce dependency on GSs for data processing. By combining edge computing with satellite technology, OEC aims to bring data processing capabilities closer to satellite sensors and communication systems. Due to the limited number of GSs and their intermittent connections with satellites, this approach increases the scalability and performance of space applications [4].

Federated Learning (FL) [5] is used in OEC for distributed machine learning among satellites to maintain privacy and reduce communication. However, traditional FL relies on GS for model aggregation, causing long training times and high power consumption. Low SWaP (Size, Weight and Power) components are necessary for highly efficient systems like OEC that need to run on limited battery and solar energy. The advancement of inter-satellite links (ISLs) [6] presents an opportunity for enhanced satellite collaboration [7], [8], yet challenges such as training time, power limitations, and accuracy persist. Previous attempts to use ISLs for decentralised aggregation encountered issues with accuracy and model staleness due to data similarity within orbits [9]. Existing approaches also suffer from satellite idleness and prolonged training times due to the need for all satellites to upload models before global updates [7], [10].

OEC encounters stringent limitations when it comes to memory and computational resources for onboard machine learning training with native floating-point representations (FP32) and commercially available hardware. The current research trends are towards narrow floating-point representations, called minifloats, that pack more operations for a given silicon area and consume less power on custom hardware such as FPGA [11]. Block minifloat (BM) has emerged as a promising solution, delivering near 32-bit floating point accuracy with 8-bit computational complexity and low memory bandwidth requirements for low-precision training at the edge [12].

In this paper, we present an extended version of our earlier work [13]. Building upon the foundations laid out in our previous research including a decentralised FL method, communication power consumption calculation, and cluster formation, this extended paper delves deeper into FL training with block minifloat (BM) arithmetic in OEC. We propose an energy-efficient version

Received 12 January 2024; revised 29 August 2024; accepted 2 October 2024. Date of publication 11 October 2024; date of current version 30 December 2024. This work was supported in part by Defence Innovation Network and in part by NSW State Government through DIN Pilot Project under Grant 2022-23. *(Corresponding author: Mohammad Reza Jabbarpour.)*

Mohammad Reza Jabbarpour, Bahman Javadi, and Rodrigo N. Calheiros are with the School of Computer, Data and Mathematical Sciences, Western Sydney University, Sydney, NSW 2000, Australia (e-mail: rjabbarpoursattari@swin.edu.au).

Philip H.W. Leong and David Boland are with the School of Electrical and Information Engineering, University of Sydney, Camperdown, NSW 2050, Australia.

Digital Object Identifier 10.1109/TSC.2024.3478768

of FedOrbit algorithm that utilises BM to accelerate FL training on custom hardware (e.g., FPGAs). Specifically, the new version of FedOrbit investigates channel parallelism and employs unified BM precision in end-to-end stochastic gradient descent (SGD)-based back-propagation Federated Learning (FL) training, featuring task-parallelism of error back-propagation and gradient generation. The use of this innovative low-precision arithmetic removes the necessity for additional computations related to layer-wise scaling factors. To the best of our knowledge, an FPGA-based FL training accelerator for OEC using 8-bit BM arithmetic is explored for the first time in this paper. In summary, this paper has the following contributions:

- It proposes FedOrbit, a novel decentralised federated learning approach by considering both intra/inter-plane ISLs in its cluster formation as well as block minifloat arithmetic in its computation.
- It proposes a general power model for OEC system by considering basic satellite functions, communication and computation power.
- It proposes a new communication strategy based on two algorithms for cluster determination using reinforcement learning and Mixed-Integer Linear Programming (MILP) based on the satellite's visiting pattern with GS.
- It presents extensive experiments on various datasets, including real satellite imagery datasets, to demonstrate the performance of FedOrbit with a significant improvement in training time and power consumption while maintaining high accuracy.

The rest of the paper is organised as follows. We discuss recent developments in the area of FL in space and low-precision and quantized FL algorithms in Section II. The system model and problem formulation are presented in Section III. The proposed FedOrbit and new clustering algorithms are presented in Section IV. Experimental results and performance evaluation are presented in Section V. Conclusion and future works are discussed in Section VI.

II. Related Work

This section discusses recent advancements in the utilisation of FL in the context of OEC. Furthermore, it investigates the current low-precision and quantized methods applied in FL with the objective of minimising power consumption.

A. FL in Orbital Edge Computing

FL is increasingly used in satellite constellations to leverage the collective intelligence of multiple interconnected satellites for improving onboard machine learning models and enhancing mission capabilities. Recent research by Chen et al. [14] showed the viability of FL for satellite training compared to GS-centric methods. However, their study used Federated Averaging (FedAvg), which is reliant on a constant GS connection. Another study introduced a taxonomy of satellite connections for FL, highlighting decentralised training's ability to address slow client convergence through predictable intermittent satellite communication [15]. The FedAvgP2P algorithm [16], an extension of FedAvg, further eliminates the necessity of a central server, including the GS.

Razmi et al. [17] introduced FedSat, assuming an ideal setup with a North Pole GS to address irregular satellite visits. Their experiments showed modifying FedAvg using satellite and GS visit patterns can effectively improve performance. FedSat reduces training time and surpasses FedAsync in convergence time and accuracy [18]; however, it is restricted to scenarios where satellites communicate with the GS once per orbital period. To extend the previous technique, FedISL [19], [20] introduced intra-satellite links, movement prediction, and partial aggregations. However, this approach involves transmitting data to a GS each round per orbit for aggregation, causing higher latency. Moreover, their servers' setup is not practical and feasible.

FedSatSchedule [21] is a scheduling FL algorithm that reduces model staleness by assessing satellite time windows. This enables downloading, local training, and uploading of the global model. Insufficient time leads to delayed model uploads for local training during long invisible intervals. FedHAP [22] substitutes the GS with high altitude platforms (HAPs) at 20-30km altitude as servers, improving performance but requiring extra hardware.

FedSpace [23] is an asynchronous FL algorithm that buffers client models, prioritising newer ones. It requires data transfer to the GS for aggregation. AsyncFLEO [9], another asynchronous algorithm, clusters satellites from distinct orbits based on data distribution similarity inferred from model weights. AsyncFLEO, like FedHAP, still depends on HAPs as additional infrastructure support.

FedLEO [8], a synchronous FL algorithm, combats slow convergence by enhancing model propagation through intra-plane and sink satellite-GS communication. Lin et al. [24] introduced a synchronous FL algorithm to tackle extended update waits in sparse satellite connections. It combines synchronous periodic and buffered aggregation strategies to adapt to changing satellite connections over time. While this approach reduces training time, drawbacks include model loss from buffered aggregation and reliance on the GS.

FedGSM [25] is another asynchronous FL algorithm that addresses the challenges of gradient staleness and learning instability by incorporating a compensation mechanism. It takes advantage of the deterministic and time-varying topology of orbits to counteract the adverse effects of staleness. Although they provide a solution for the staleness problem, they only consider time-dependent compensation in their FL frameworks.

Lin et al. [26] proposed FedSN framework for LEO satellite networks aiming to enhance model training effectiveness. FedSN includes a sub-structure scheme addressing resource limitations, training imbalances, and intra-group model staleness. To minimise the communication rounds, an approach named LEOShot [27] has been introduced. This one-shot FL technique aims to attain model training convergence within a single communication round.

The LEO Edge Selection and Clustering (LESC) approach, proposed by Chen et al. [10], combines FL, edge server selection and client clustering techniques to improve performance, latency, and quality of service in LEO-based networks. However, because it considers the connection between the LEO edge server

TABLE I
COMPARISON OF THE STATE OF THE ART FL APPROACHES IN SPACE

Technique	Ref.	Server	Communication	Metrics	Approach
-	[14]	GS/LEO	Intra-plane ISLs	Communication cost & Latency	Investigated 4 possible learning strategies in SatCom
-	[15]	GS	Inter/intra-plane ISLs	Convergence & Accuracy	Inter-plane ISLs are used in satellite FL
FedSat	[17]	GS	No ISL	Convergence & Accuracy	Introduced satellite FL and applied FedAvg
FedISL*	[19]	GS/MEO	Intra-plane ISLs	Training time & communication cost	Used satellite predictability and partial aggregation
FedSatSchedule	[21]	GS	No ISL	Convergence & model staleness	Developed a scheduler to consider visit duration predictability
FedHAp*	[22]	HAP	Intra-plane ISLs, inter-HAP	Convergence & Training Time	Integrated HAPs into a synchronous FL framework as servers.
FedSpace	[23]	GS	No ISL	Training time, staleness & idleness	Incorporated heuristic GS update procedure and gradient buffering
AsyncFLEO*	[9]	HAP	Intra-plane ISLs, inter-HAP	Convergence & Accuracy	Leveraged FedSyn to tackle straggler satellites and model staleness
FedLEO*	[8]	GS, Sink Satellite	Intra-plane ISLs	Convergence & Accuracy	Used the predictability of satellite orbiting patterns for fast convergence.
-	[24]	GS	No ISLs	Convergence & Accuracy	Integrated two aggregation strategies to address the variations in satellite connections' heterogeneity over time.
FedGSM	[25]	GS	No ISLs	Accuracy & model staleness	Leveraged the deterministic and time-varying topology of the orbits
FedSN	[26]	GS	No ISLs	Accuracy, Communication & computing overhead	Utilised two components: sub-structure scheme and pseudo-synchronous model aggregation.
LEOShot	[27]	GS	Intra-plane ISLs	Convergence & Accuracy	Comprised three processes: synthetic data generation, knowledge distillation, and virtual model retraining
FELLO*	[10]	GS/LEO	Inter/intra-plane ISLs	Latency & Accuracy	Combined FL, edge server selection and client clustering techniques.
DSFL	[7]	GS	Inter/intra-plane ISLs	Training time & communication cost	Proposed to find the efficient path among planes for model sharing.
FedOrbit*	This Paper	GS,LEO	Inter/intra-plane ISLs	Accuracy, Training time & power consumption	Novel cluster formation based on the visiting patterns, inter/intra-plane ISLs and reinforcement learning, and low-precision arithmetic

* Cluster-based FL approaches

and the GS for server selection, the approach limits the number of LEO edge servers and most of the satellites will not participate in the training procedure.

Decentralised Satellite Federated Learning (DSFL) [7] is an approach that utilises both intra and inter-plane transmissions in its proposed algorithm. Also, an energy-aware communication strategy (EACS) is utilised to find the efficient path among planes for model sharing. DSFL uses intra and inter-plane communications for global aggregation, which leads to high aggregation time and cost.

To summarise, while many existing approaches have primarily addressed concerns regarding satellite idleness and model staleness, there is a significant gap in the literature when it comes to studying the crucial parameters of training time and power consumption in OEC. Table I listed a summary of the existing work as well as our proposed FedOrbit technique to address the training time and power consumption where cluster-based FL approaches are starred. In addition, as represented in Table I, existing approaches can be categorised into 4 categories based on their communication links, namely No ISL, Intra-plane ISLs, Inter-HAP, and Inter/intra-plane ISLs.

B. Low-Precision and Quantised FL Methods

Transmitting a large number of updated model parameters over the communication channel with limited throughput from the clients (users) to the server is one of the main challenges of FL [28]. This challenge can be tackled by reducing the number of participating users, via, e.g., scheduling policies [29]. Another approach is to reduce the data volume transmitted by each user, achieved through techniques like sparsification or scalar quantisation. For example, various approaches to condense the updates transmitted from users to the server are proposed by Konečnỳ et al. [30]. These techniques encompass random masks, sub-sampling, and probabilistic quantisation. The utilisation of sparsifying masks for compressing gradients was proposed by Hardy et al. [31]. Sparse ternary compression (STC), a new compression framework that is specifically designed to meet the requirements of the FL environment was developed by Sattler et al. [32]. Additional variations of probabilistic scalar quantisation for FL were explored by many researchers [33], [34], [35]. However, these methods are considered sub-optimal from a quantisation theory point of view.

This motivates the exploration and evaluation of quantisation techniques to streamline the transfer of updated models in FL. The goal is to minimise the error induced by quantisation in the aggregated global model. In this context, the UVeQFed algorithm [36] extends scalar quantisation to vector quantisation and uses lossless compression via arithmetic coding. Li et al. [37] introduced a quantized FL algorithm for specific loss functions, applying stochastic quantisation to model parameters instead of local gradients, finding that while convergence rates are unaffected, error bounds increase with quantisation. Hönig et al. [38] proposed DAdaQuant, a doubly adaptive quantisation algorithm that dynamically adjusts quantisation levels over time and across different clients, improving compression without compromising model quality.

Although existing gradient compression algorithms are mainly designed for data centres with high-speed networks, a general framework, called hyper-sphere quantisation (HSQ), was proposed by Dai et al. [39]. HSQ can be configured to achieve a continuum of trade-offs between communication efficiency and gradient accuracy. Most of the existing methods assume homogeneous quantisation for all clients, disregarding the real-world scenario where devices exhibit heterogeneity and support varying levels of quantisation precision. Federated Learning with Heterogeneous Quantisation (FEDHQ) [40], assigns distinct weights to clients by minimising the convergence rate upper bound, which is dependent on the quantisation errors of all clients. A new method called QuAsyncFL [41] integrates asynchronous FL with an unbiased nonuniform quantiser to tackle the challenge of inadequate communication efficiency.

Ji et al. [42] introduced FedQNN, a FL framework designed to reduce computational complexity and energy consumption

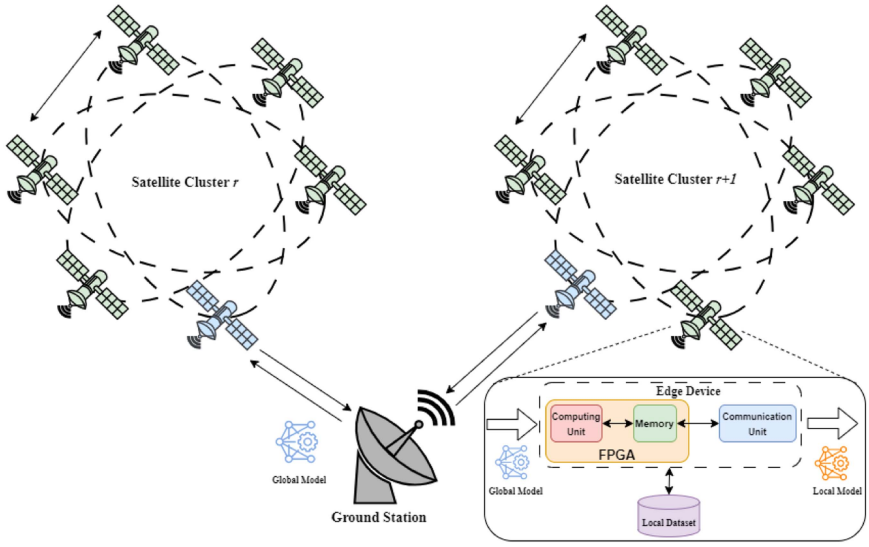


Fig. 1. Orbital Edge Computing Architecture using FPGA.

for IoT devices. It employs ultra-low bit-width quantisation to minimised energy use and incorporates sparse ternary and binary mask compression to reduce data transmission. Additionally, Aketi et al. [43] proposed a low-precision decentralised training approach in a peer-to-peer setup to further mitigate computational and communication costs.

In summary, most existing low-precision and quantized FL methods have focused on reducing communication costs while overlooking computational costs. Additionally, there is a lack of research on applying compute-efficient training techniques, such as quantisation and low-precision methods, specifically for FL in OEC setups. Addressing these gaps is one of the significant contribution of this paper.

III. System Model and Problem Formulation

The system architecture for orbital edge computing for satellite constellations is outlined in this section as depicted in Fig. 1. Satellites establish clusters with neighbours based on ISL range, considering both intra-plane and inter-plane ISLs and the selection process of master satellites for GS communication (i.e., blue colour in Fig. 1). Each satellite is equipped with an edge device which includes a communication unit and an FPGA board containing computing and memory units to perform training in single-precision (FP32) and low-precision (BM) formats. The proposed FL approach involves four phases: model distribution, clustering, local training, and aggregation. Initially, one satellite per orbit receives and disseminates models from the GS to its intra-orbit neighbours, based on the method proposed by Razmi et al. [19]. Once distributed, satellites move to the subsequent phase. To improve readability, important notations are summarised in Table II.

Satellites are partitioned into r clusters where satellite $i \in \{1, \ldots, m\}$ in orbit o collects and stores a dataset $D_{o,i}$. This local data set is used to train an ML model via the Federating Average peer-to-peer (FedAvgP2P) algorithm [16]. Each cluster, C_r, includes a set of satellites in which the training process is orchestrated by one of the cluster's satellites. This satellite is called the master satellite and it sends a request to other satellites within its ISL range at round t, $K^t_{o,i}$, to form a cluster and get their updated weights for aggregation.

Once the parameter distribution and clustering phases are completed, all satellites proceed to initiate their local training and aggregation phases using the computation model described in Section III-A. Subsequently, the master satellites transfer their updated weights to the GS when the next opportunity to communicate occurs. The GS then carries out the global aggregation phase by incorporating the received updated weights from the master satellites and broadcasts the new global model to the satellites at the next communication interval.

A satellite constellation with a different number of orbit planes (OPs) and satellites is considered in this paper. We assign a unique ID to each satellite based on its OP number. Orbit set is defined as $O = \{o\}$ where $o = \{1, 2, \ldots, n\}$. Similarly, the satellite set is defined as $S = \{s_{o,i}\}$ where $i = \{1, 2, \ldots, m\}$. In other words, each orbit o contains m equally spaced satellites with unique IDs $\{s_{o,1}, s_{o,2}, \ldots, s_{o,m}\}$. The base speed of each satellite in orbit o, v_o, is calculated as

$$v_o = \sqrt{\frac{\mu}{A_o + r_E}} \tag{1}$$

where r_E is the earth radius (6371km), μ is the geocentric gravitational constant ($3.98 \times 10^{14} m^3/s^2$) and A_o is orbit altitude.

The orbit period of the satellite in orbit o is computed as follows:

$$P_o = \frac{2\pi(A_o + r_E)}{v_o} \tag{2}$$

A. Computation Model

Satellites collaboratively learn a global model in cluster C_r by minimising a global objective function as follows:

$$w^* = \min F_r(w) \tag{3}$$

TABLE II
IMPORTANT NOTATION

Notation	Meaning
o	Orbit plane ID
i	Satellite ID
$S_{o,i}$	Satellite i in Orbit o
r	Number of Clusters
C_r	r^{th} cluster
$D_{o,i}$	Size of local dateset of satellite i in Orbit o
D	Size of the whole dataset
t	Round Number
w	model weight
x	batch size (data point)
η	Learning rate
$F_r(w)$	Global loss function of satellites in cluster C_r
$f_{o,i}(w)$	Local loss function of satellite i in Orbit o
$w^t_{o,i}$	Model weight of satellite i in Orbit o and round t
$N^t_{o,i}$	Randomly selected Cluster Neighbours of satellite i in Orbit o and round t
$K^t_{o,i}$	Cluster Neighbours of satellite i in Orbit o and round t
k	Size of aggregation neighbour list
m	Cluster size
f	fraction Coefficient
v_o	Orbit Base Speed
r_E	Earth Radius
μ	Geocentric Gravitational Constant
A_o	Orbit Altitude
P_o	orbit period of the satellite in orbit o
$E((o_a,i),(o_b,j))$	Euclidean distance between satellites $S_{o_a,i}$ and $S_{o_b,j}$
$E_T((o_a,i),(o_b,j))$	threshold value
$f_{CPU/GPU_{(o,i)}}$	number of CPU/GPU cycles per second
$c_{o,i}$	required number of CPU/GPU cycles to process 1 bit of data
α_e	minimum elevation angle
$R_{o,i}$	traffic flow through the i^{th} satellite on o^{th} orbit
$P^{TP}_{o,i}$	basic power of satellite laser terminal
$P^{AP}_{o,i}$	basic power of microwave antenna
$P_{o,i}$	Power consumption of satellite i in orbit o
$T_{o,i}$	Computing time of satellite i in orbit o
$E_{o,i}$	Energy consumption of satellite i in orbit o
$C_{o,i}$	Communication cost of satellite i in orbit o
z	effective capacitance coefficient of the processor

The global loss function on all the distributed datasets is:

$$F_r(w) = \sum_{\substack{i\in S\\ o\in O}} \frac{D_{(o,i)_r}}{D_r} f_{o,i}(w) \tag{4}$$

where $D_{o,i} = |D_{o,i}|$, $D_r = \sum_{\substack{i\in S\\ o\in O}} D_{(o,i)_r}$ is the size of the whole dataset in the cluster, w is the model parameter, and $f_{o,i}(w)$ is the local loss function that is computed for each dataset $D_{o,i}$ as follows:

$$f_{o,i}(w) = \frac{1}{D_{o,i}} \sum_{x\in D_{o,i}} f_{o,i}(w;x) \tag{5}$$

where $f_{o,i}(w;x)$ is training loss for a data point x and model parameter w.

Each satellite receives its own model via the first phase and can communicate with its cluster master satellite via ISLs. All satellites initialise their training models with the same initial weight w_0. At every round t, each satellite $S_{o,i}$ trains the model by carrying out Stochastic Gradient Descent (SGD) steps over its own dataset $D_{o,i}$, resulting in a model weight $w^t_{o,i}$, and updates the local version of the global model parameters as

$$w^{t+1}_{o,i} = w^t_{o,i} - \eta \nabla f_{o,i}\left(w^t_{o,i}\right) \tag{6}$$

where η is the learning rate. All satellites in a cluster will send their updated weights to the master satellite. Then, the master satellite aggregates the neighbour models with its local model as follows:

$$w^{t+1}_{o,i} = w^t_{o,i} - \eta \nabla f_{o,i}\left(w^t_{o,i}\right) + \sum_{i\in N^t_{o,i}} \left(w^t_{o,i} - \eta \nabla f_{o,i}\left(w^t_{o,i}\right)\right) \tag{7}$$

where $N^t_{o,i}$ is the round t's neighbours of satellite $S_{o,i}$, randomly sampled from the master satellite's cluster neighbours set, $K^t_{o,i}$, $(N^t_{o,i} \subseteq K^t_{o,i}, |N^t_{o,i}| = k, |K^t_{o,i}| = m, k \le m, k = m \times f$, where f is the fraction of neighbours (e.g., 0.6).

Finally, GS receives the local updates $w^{t+1}_{o,i}$ from all master satellites, and aggregates them into new global model parameters as follows:

$$w^{t+1} = \frac{1}{r} \sum_{\substack{i\in S\\ o\in O}} \frac{D_{o,i}}{D} w^t_{o,i} \tag{8}$$

GS then advances into the next aggregation round.

B. Communication Model

This paper assumes free space optics (FSO) communication links among satellites as a means to overcome intermittent connectivity between satellites and GS. An FSO link is utilised for transmitting optical signals between transmitters and receivers over the atmosphere or the vacuum environment in space, in a clear line of sight condition [44]. These links can be divided into four main types, namely terrestrial, aerial (non-terrestrial), space, and deep-space. FSO links between satellites are examples of space FSO links referred to as Laser Inter-Satellite Links (LISLs) in the rest of the paper.

Laser links are considered in this work due to their advantages over Radio Frequency (RF) links. Higher data rates, smaller antenna size, lesser weight, volume, and interference, narrower beams, higher security, and lower transmission power are some of these advantages [45]. LISLs can be divided into two types, namely intra-orbital plane LISL, which indicates the communication between satellites from the same OP, and inter-orbital plane LISL, which indicates the communication between satellites from different OPs [6]. The latter case can be further classified into 3 types: adjacent OP LISL (AOPL), nearby OP LISL (NOPL) and crossing OP LISL (COPL).

Although the altitude of different OPs is the same and satellites in these OPs move at the same velocity, the direction of satellites in adjacent and nearby OPs is slightly different, which leads to different relative speeds of satellites in these OPs. LISLs can be further divided into permanent and temporary based on the duration of communication between satellites. Intra-orbital plane LISL, AOPL, and NOPL are usually permanent, while

COPL is a temporary communication. Permanent LISLs are easy to establish and maintain due to their stability and long duration. Hence, these types of LISLs are considered in this paper. The effect of varying the range of a satellite's LISLs on the number of permanent LISLs was studied by Chaudhry and Yanikomeroglu [6].

By taking into account terminals, computational and power constraints, LISL range can be different for any satellite. Hence, the satellite cluster size could be in the range between 2 and 88 as presented by Chaudhry and Yanikomeroglu [6]. Satellites in each orbit can create a ring network via LISLs.

As mentioned above, a clear line of sight between satellites is required for inter/intra-satellite communication. By taking into account geometric considerations, it can be shown that satellites $s_{o_a,i}$ and $s_{o_b,j}$ have a line of sight view if $E((o_a,i),(o_b,j)) < E_T((o_a,i),(o_b,j)) = \sqrt{(A_{(o_a,i)}+r_E)^2 - r_E^2} + \sqrt{(A_{(o_b,j)}+r_E)^2 - r_E^2}$, where $E((o_a,i),(o_b,j))$ and $E_T((o_a,i),(o_b,j))$ are the Euclidean distance between satellites $s_{o_a,i}$ and $s_{o_b,j}$, and threshold value, respectively.

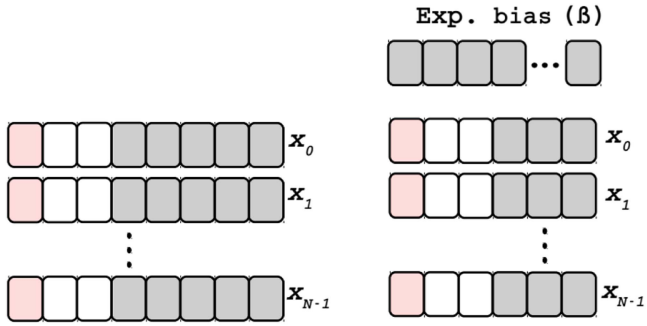
In our proposed technique, GS communication is only required at the initialisation phase and final aggregation at each training step, where master satellites send their clusters' local updates to GS for final aggregation. For this type of communication, the satellite and GS should be visible to each other. By considering the minimum elevation angle α_e, the satellite $s_{o_a,i}$ and GS can visit each other if $\frac{\pi}{2} - \angle(p_{GS}, p_{(o_a,i)} - p_{GS}) \geq \alpha_e$, where p_{GS} and $p_{(o_a,i)}$ indicate the position of GS and satellite $s_{o_a,i}$, respectively. As mentioned before, our proposed approach is designed to handle intermittent connectivity between satellites and the GS. This technique utilises both intra-OP LISLs and inter-OP LISLs to reach this goal.

C. Block MiniFloat Arithmetic

Block minifloat (BM) arithmetic is a specialised approach to representing and processing floating-point numbers using fixed-size blocks or tiles [12]. Unlike traditional floating-point formats, block minifloats divide the bit pattern into distinct sections, often including components such as sign bits, exponents, and mantissas. This compact representation is designed to balance precision and resource efficiency, making it particularly relevant in contexts where computational resources are constrained. BM arithmetic is gaining attention for its ability to perform arithmetic operations with reduced bitwidths, leading to more efficient implementations in various applications, such as machine learning and embedded systems [11].

The actual value of a minifloat number, with (e, m) indicating the quantity of exponent and mantissa bits in its encoding is calculated as follows:

$$X\langle e,m\rangle = \begin{cases} E=0, & (-1)^s \times 2^{1-\beta} \times (0 + F \times 2^{-m}) \quad \text{(denormal)} \\ \text{(otherwise,)} & (-1)^s \times 2^{E-\beta} \times (0 + F \times 2^{-m}) \quad \text{(normal)} \end{cases} \tag{9}$$



(a) Minifloat. (b) BM (Shared exp. bias).

Fig. 2. Minifloat and BM tensor representations.

The exponent and mantissa in their decimal representations are non-negative integers, denoted as E and F, while s represents the sign bit, and $\beta = 2^{e-1}$, serves as the bias for the binary offset encoding method. These characteristics align with IEEE-754 floating point standards, with the distinction that minifloats are notably smaller, typically consisting of only 4-8 bits. The key distinction between minifloat and BM representations are depicted in Fig. 2.

BM utilises an 8-bit common exponent for every group of tiles called a block. Each block is a BM tensor a, containing $T \times T$ minifloats, where T is the tile size.[1] The shared exponent, β_a, functions as a scaling factor specific to a, guaranteeing that the highest 8-bit minifloat corresponds with the maximum floating point value. Thus the value of the i-th element of a, is

$$a_i = X_i\langle e,m\rangle \times 2^{-\beta_a} \tag{10}$$

and we set

$$\beta_a = \max\left(\lfloor \log_2 |a| \rfloor\right) - (2^e - 1) \tag{11}$$

The first term denotes the maximum exponent for the tensor a, which changes and must be updated during training, while the second term is fixed and refers to the maximum exponent of X. This strategy prevents overflow and offers more precise scaling than conventional layer-wise loss/gradient scaling techniques.

The dot product between two BM vectors, v_a and v_b is computed as

$$v_a.v_b = \sum_{i=1}^{N} \left(\left(X_i^{V_a} \times 2^{-\beta_{V_a}} \right) \times \left(X_i^{V_b} \times 2^{-\beta_{V_b}} \right) \right) \tag{12}$$

$$= 2^{-\left(\beta_{V_a} + \beta_{V_b}\right)} \left(X_{V_a}.X_{V_b} \right). \tag{13}$$

It is important to note that (13) is computed as a mostly dense minifloat dot product, improving efficiency. Multiplication of minifloats follows the semantics of conventional IEEE 754 floating point arithmetic, including gradual underflow [46]. Infinity and NaN are not supported.

Matrix-vector and matrix-matrix operations are computed as repeated dot products according to (13). For inference and training, all weight, activation and gradient tensors are quantised with

[1] In this paper, the value of T is set to 48

stochastic rounding to BM format, with 48×48 BM tiles. The specific training algorithms proposed in this paper are described in the Section IV. More details about BM number system and architecture can be found in the original paper by Fox et al. [12].

D. Power Consumption Model

The power consumption of laser-based communication satellites can be divided into three distinct components. The first segment is related to the consumption of basic functions $P_{o,i}^{base}$, which encompass activities such as maintaining the satellite's altitude and rotating its solar panels. The second segment, communication power consumption, $P_{o,i}^{Comm}$, which includes the consumption of laser terminals $P_{o,i}^{LISL}$, is used to establish LISLs and the power consumption of microwave communication antennas, $P_{o,i}^{GS2S}$, is utilised to create links between GSs and satellites. The third segment is the power consumption of the on-board processor for processing the requested task (i.e., training the model), $P_{o,i}^{Comp}$. The overall power consumption is calculated as follows:

$$P_{o,i} = P_{o,i}^{base} + P_{o,i}^{Comm} + P_{o,i}^{Comp} \tag{14}$$

The primary focus of this study is on communication and computation power consumption, therefore, the basic power consumption is considered as a constant. $R_{o,i}$ is considered as the traffic flow (data rate) through the satellite $S_{o,i}$ and computed as follows:

$$R_{o,i} = \sum R_{o,i}^{LISL} + \sum R_{o,i}^{GS2S} \tag{15}$$

$R_{o,i}^{LISL}$ and $R_{o,i}^{GS2S}$ are data rates between satellites and GS, respectively. The communication power consumption of satellites is computed as follows:

$$P_{o,i}^{Comm} = P_{o,i}^{LISL} + P_{o,i}^{GS2S} \tag{16}$$

where $P_{o,i}^{LISL}$ and $P_{o,i}^{GS2S}$ are computed as:

$$P_{o,i}^{LISL} = P_{o,i}^{TP} + a \sum R_{o,i}^{LISL} \tag{17}$$

$$P_{o,i}^{GS2S} = P_{o,i}^{AP} + b \sum R_{o,i}^{GS2S} \tag{18}$$

where $P_{o,i}^{TP}$ and $P_{o,i}^{AP}$ are the basic power of the satellite laser terminal and the microwave antenna, respectively. Two parameters $a(=1)$ and $b(=5)$ are constant coefficients expressed in Watt/Mbps [47].

Regarding the computation power consumption of satellites as a result of a training task, a model that captures the most significant CPU and GPU parameters is considered [48], [49]. The power consumed during processing (computing) at the i-th satellite in o-th LEO orbit is defined in this model as:

$$P_{o,i}^{Comp} = S(D_{o,i}) \times c_{o,i} \times f_{CPU/GPU_{(o,i)}}^2 \times z \tag{19}$$

where $c_{o,i}$ is the required number of CPU/GPU cycles to process 1 b of data, $S(D_{o,i})$ is the size of its dataset in bits, $f_{CPU/GPU_{(o,i)}}$ the CPU/GPU frequency (the number of CPU/GPU cycles per second), and z is the effective capacitance coefficient of the processor.

Regarding the computing energy consumption of satellites as a result of a training task, an estimation model that considers the computing time and consumed power is considered in this study. The energy consumed during computing at the i-th satellite in o-th LEO orbit is defined in this model as:

$$E_{o,i}^{Comp} = T_{o,i}^{Comp} \times P_{o,i}^{Comp} \tag{20}$$

where $T_{o,i}^{Comp}$ and $P_{o,i}^{Comp}$ are the computing time and consumed power, respectively.

By considering that within each epoch, the forward and backward passes are executed for each batch of data, and the number of forward and backward paths per epoch depends on the size of the dataset and batch size, computing time for each epoch is calculated as follows:

$$T_{o,i}^{Comp_{epoch}} = \frac{\text{Size of training samples}}{\text{Batch size}} \times (\text{FP+BP}) \tag{21}$$

where FP and BP are the number of forward and backward passes which are executed for each batch of data.

Despite extensive research on FL, most existing approaches heavily depend on the GS for model aggregation. This dependency not only imposes significant communication overhead but also exacerbates the limitations of edge resources, such as constrained power and computational capacity. These challenges highlight a critical gap in current methodologies: the need to optimize communication between satellites and the GS to reduce both power consumption and training time. This paper aims to address this problem by proposing a novel energy-efficient and decentralised cluster-based FL for optimizing the communication process in FL systems, thereby enhancing overall efficiency and sustainability.

IV. The Proposed FedOrbit Algorithm

In this section, the proposed on-board federated learning approach called FedOrbit and its cluster formation mechanisms are discussed.

A. FedOrbit

FedOrbit considers intra-plane and inter-plane ISLs as well as satellite visiting patterns for cluster formation [13]. The proposed algorithm has four phases including parameter distribution, clustering, local training, and aggregation which, are explained in Algorithm 1. After the initial parameter distribution phase, FedOrbit will perform the cluster formations (line 4) which can be done by two new algorithms for master satellite selection (20 and Algorithm 2). Cluster formation is discussed in more details in Section IV-B. After master satellite selection and cluster formation, all satellites initialise their local training and aggregation phases (Lines 5–28) based on the described computation model in Sections III-A and III-C. Selection of BM or FP32 format for local training is made in Lines 5–9 before local training phase. Finally, in the global aggregation phase (Lines 29–39), master satellites send their updated weights to the GS upon their next visit (Lines 30–38). GS execute the global aggregation phase based on the updated weights and broadcasts the new global model to satellites in their upcoming visits (line 39).

B. Cluster Formation

Cluster formation is one of the main steps in FedOrbit and is formulated as an optimisation problem in this paper. Two approaches for solving the problem are investigated, in which minimising communication cost and maximising model accuracy are considered the objective functions, respectively. In the first approach, Mixed-Integer Linear Programming (MILP) is utilised to construct a formation (allocation) strategy to reduce the satellites' communication cost under the constraints of satellite communication range and number of clusters. As mentioned before, satellites are partitioned into c clusters where satellite $i \in \{1, \ldots, m\}$ in orbit o collects and stores a dataset $D_{o,i}$. Each cluster, C_r, includes a set of satellites in which the training process is orchestrated by one of the cluster's satellites. Communication cost is defined as follows:

$$C_{o,i}^{Comm} = c_1 \left(C_{o,i}^{LISL}\right) + c_2 \left(C_{o,i}^{GS2S}\right) \tag{22}$$

where c_1 and c_2 are constant values based on the LISL/GS bandwidth, data rate and training round, $C_{o,i}^{LISL}$ is the cost (number of satellites that need to connect to each other) of intra/inter satellites communications, and $C_{o,i}^{GS2S}$ is the cost (number of satellites that need to connect to GS) of master satellites and GS communications. $C_{o,i}^{LISL}$ and $C_{o,i}^{GS2S}$ can be determined based on cluster size and number of clusters, respectively. Hence, communication cost can be considered as:

$$C = c_1 \sum_{i=1}^{S} x_i + c_2 x_j \tag{23}$$

The optimisation function can be formulated as:

$$\begin{aligned} \text{minimize} \quad & C \\ \text{subject to} \quad & \sum_{i=1}^{S} x_i = S, \\ & 0 \leqslant x_i \leqslant 11, \\ & x_j = Sum\left[double\left(\left(\sum_{i=1}^{S} x_i\right) > 0\right)\right], \\ & \frac{S}{11} \leqslant x_j \leqslant \frac{S}{3} \end{aligned} \tag{24}$$

where S is the number of the desired satellites, x_i indicates the satellite $i \in \{1, 2, \ldots, S\}$ and based on the first and second constraints its value should be between 0 to 11 (by considering ISL ranges discussed in [6]) and the sum of the selected satellites should be equal to the number of desired satellites. The third constraint indicates the number of clusters (based on the number of selected satellites and their neighbours (cluster size)). The number of clusters is determined by the fourth constraint. Cluster formation can be obtained by solving this optimisation problem where the selected x_i satellites will act as master satellites.

In the second cluster formation (Algorithm 2), the master satellites are selected based on two main procedures: cluster determination via Reinforcement Learning (RL) and visiting patterns of satellites with GS. In general, satellites have intermittent but predictable connectivity to the GS as a result of their orbital movement. Although the visiting pattern is time-varying, it is deterministic and can be computed based, for example, on the method proposed by Ali et al. [50]. Hence, in the *FedOrbit* cluster formation algorithm, master satellites are selected from satellites that will visit the GS after their cluster training is finished. At line 1, if the visiting time of a satellite with the GS falls within the "training time" and "training time plus

Algorithm 1: FedOrbit Algorithm.

▷ Parameter Distribution Phase
1: **initialise** epoch n = E, satellite ID i, orbit plane ID o, w_0, q, f, T, e, m
2: **for** each state s from 1 to T **do**
3: **for** each round t = 1,2,... **do**
4: Cluster formation based on (20) **or Algo. 2** ▷ Clustering Phase
5: **if** BM = True **then**
6: set the weights, activations, errors, and momentum based on BM(e,m)
7: **else**
8: set the weights, activations, errors, and momentum based on FP32(8,23)
9: **end if**
10: **All Satellite in cluster execute:** ▷ Local Training Phase
11: **Function** LocalTraining $(S_{o,i}, w)$
12: x = (split $D_{o,i}$ into batches of size X)
13: $v = |D_{o,i}|$
14: **for** each local epoch n from 1 to E **do**
15: **for** each batch $x \in X$ **do**
16: $w_{o,i}^{t+1} = w_{o,i}^{t} - \eta \nabla f_{o,i}(w_{o,i}^{t}; x)$
17: **end for**
18: **return** w and v to master Satellite
19: **end for**
20: **Master Satellite in cluster executes:** ▷ Local Aggregation Phase
21: $k = max\{\lceil m \times f \rceil, 2\}$
22: $N_{o,i}^{t}$ = (random set of k neighbours in cluster)
23: **for** each neighbour $S_{o,i} \in N_{o,i}^{t}$ in parallel **do**
24: $w_{o,i}^{t} = LocalTraining(S_{o,i}, w)$
25: **end for**
26: $w_{o,i}^{t+1} = w_{o,i}^{t} - \eta \nabla f_{o,i}(w_{o,i}^{t}) + \sum_{j \in N_{o,i}^{t}} (w_{o,j}^{t} - \eta \nabla f_{o,j}(w_{o,j}^{t}))$
27: Set M = $\emptyset$, $w^t = 0$
28: **end for**
29: **Ground Station executes:** ▷ Global Aggregation Phase
30: **while** i < len(masterNodeSet) **do**
31: Upon incoming connection by master satellite $S_{o,i}$:
32: **if** $S_{o,i} \notin M$ and received weights $w_{o,i}^{t}$ **then**
33: Update $w^t = w^t + w_{o,i}^{t}$
34: Add $S_{o,i}$ to M
35: **else**
36: Terminate Connection
37: **end if**
38: **end while**
39: Update & Broadcast $w^{t+1} = \sum_{\substack{i \in S \\ o \in O}} \frac{D_{o,i}}{D} w_{o,i}^{t}$
40: **end for**

Algorithm 2: FedOrbit Cluster Formation.

1: CandicateMasterNodeSet = estimated local training time ≤ Satellites-GS visit time GS ≤ estimated local training time + predefined threshold
2: masterNodeSet = Satellites ∈ CandicateMasterNodeSet that fit to **determined cluster formation based on Algorithm 3**
3: **for** each satellite $S_{o,i}$ ∈ masterNodeSet **do**
4: **Send** neighbour request based on $O_{o,i}$
5: **Wait** for t(s) to **receive** neighbour reply
6: **Create** $K^t_{o,i}$ based on received replies
7: $m = |K^t_{o,i}|$
8: **end for**

Algorithm 3: FedOrbit Cluster Determination.

1: **_build_model(self):**
2: RL model = Conv2D(64, (2, 2)), Flatten(), Dense(128)
3: **return** created model
4: **for** episode in range(number_of_episodes) **do**
5: **for** time_step in range(number_of_steps) **do**
6: action = agent.act(state)
7: next_state, reward, accuracy = env.step(action)
8: reward += $100 \times \frac{accuracy}{((time\_step+1)\times cluster\_size)}$
9: **return** reward
10: **end for**
11: **end for**
12: **return** determined cluster formation

predefined threshold", it will be selected as a master satellite candidate. Training time is predicted based on the utilised model and dataset. Predefined threshold is a time variable defined in minutes. When multiple satellites meet this condition simultaneously, the satellite that visits the GS sooner is selected. If no satellite meets the criteria within the initial time range, the "predefined threshold" will be incrementally increased until at least one satellite qualifies. In line 2, a number of candidate satellites are chosen as master satellites based on the desired scenario and the cluster formation determined by Algorithm 3. In Lines 3–8, the master satellites create their clusters based on sent and received responses from neighbours.

To find the best cluster formation using the presented optimisation formulation, Algorithm 3 is proposed based on an RL algorithm. Initial state is randomly created based on the number of selected master satellites, their communication ranges and total number of desired satellites. The possible actions include adding/removing a cluster, adding/removing satellites from cluster, and doing nothing. IF the agent select an action which is not feasible (e.g., adding a satellite to a cluster more than its capacity), it will receive penalty (negative reward). The Reward function based on accuracy, cluster size, and number of steps is calculated after taking an action in a state to guide the agent towards desirable behaviour (i.e., finding the best formation with high accuracy within the lower number of steps (attempts)). As Lines 1–3 show, this algorithm uses deep Q-learning with a 2×2 convolution layer 64 channel with ReLu activation, followed by a flatten and two fully connected layers with 128 units to build a model. It uses Adaptive Moment Estimation (Adam) as an optimiser with the following parameters: gamma (discounting rate) 0.95, epsilon (exploration rate) 1.0, epsilon min 0.01, epsilon decay 0.995 and learning rate = 0.03. Through Lines 4–12, the model trains based on the defined reward function (formulated by considering accuracy, number of steps and cluster size in line 8) and environment and return determined cluster formation to be used in Algorithm 3.

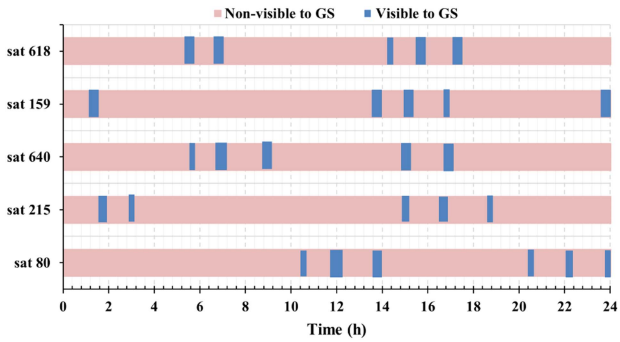


Fig. 3. Visiting patterns of 5 satellites and a GS in 24 hours.

For more clarification, the visiting pattern between 5 satellites and the GS in Canberra, Australia in 24 hours is depicted in Fig. 3. The visiting patterns of satellites vary due to orbit specifications and the movements of both the satellites and the Earth, as depicted in this figure. Satellites from different orbits with the following IDs, Sat 80 ($S_{4,20}$), Sat 215 ($S_{11,15}$), Sat 640 ($S_{32,20}$), Sat 159 ($S_{8,19}$) and Sat 618 ($S_{31,18}$), are plotted in this example. Satellites 215 and 159 have their first visit within the next 2 hours. However, satellite 80 has the first visit after 10.5 hours. It should be noted that on average each satellite visits the GS 5 times during 24 hours in our satellite constellation setting.

V. Performance Evaluation

Experiments on various models and datasets including real satellite imagery datasets were performed to demonstrate the performance of the proposed FedOrbit algorithm. The model accuracy, power consumption and training time are the evaluation metrics of interest in this work. We developed the FedOrbit algorithm in the FedML framework [51] for FL implementation. The Satellite Communications Toolbox from Matlab is utilised to compute the visiting pattern of LEO satellites with regard to the GS. We used m5.12xlarge Amazon EC2 instances (includes 48 vCPUs, 192.0 GiB memory) for most of our experiments. It is worth noting that the real-time power consumption is measured and utilised via built-in commands (powertop and nvidia-smi) for experiments in this paper.

A. Experimental Setup

Satellite Constellation: Table III lists the experimental parameters in this paper. A Walker-Delta constellation (similar to Starlink) consisting of 720 LEO satellites with an inclination of 70°, altitude of 570km, 36 orbital planes (OPs), and 20 satellites

TABLE III
EXPERIMENTAL PARAMETERS

Parameter	Value
Constellation	Walker Delta (Starlink)
Number of LEO satellites	720
Number of orbits	36
Number of LEOs per Orbit	20
Inclination	70°
Altitude	570 km
Bandwidth of LISL/GS2S	2.5/1.25 GHz
System Loss	3 dB
Frequency	27 GHz
Gain to noise temperature ratio	5 dB/K
Data rate	16 Mbps
Power	40 Watt
Total FL rounds	40
Local training epochs	30
Batch size	10

in each OP is considered. The spacing between OPs and satellites in each OP are $10°(i.e., 360°/36)$ and $18°(i.e., 360°/20)$, respectively. Each satellite has a unique ID based on its own ID and OP number. In other words, 36 orbits contain 20 equally spaced satellites with 720 unique IDs $\{S_{0,0}, S_{0,1}, \ldots, S_{35,18}, S_{35,19}\}$. Moreover, a GS is located in Canberra, Australia with Latitude (-35.40139) and Longitude (148.98167). It is worth noting that the LISL range of satellites $\in \{659, 1319, 1500, 1700\}$ km allows the master satellite to create a cluster with 2,4,6,10 neighbours.

Baselines: Our proposed FedOrbit is compared with a number of the existing baseline approaches, including FedSyn [5], FedLEO [8], and FELLO [10]. FedSyn is the traditional version of FL where local models are sent to a central server (GS) for aggregation. FedLEO enhances the traditional FL star topology by using intra-plane communication pathways, facilitating model propagation among satellites. Additionally, it optimises the scheduling of communication between "sink" (master) satellites and the GS by leveraging the predictability inherent in satellite orbiting patterns. In FELLO, the GS initially identifies an LEO satellite with superior communication link quality to serve as an FL edge server. Subsequently, this LEO server employs a clustering mechanism and evaluates communication capability via optical ISLs to determine its LEO clients. FedSyn is selected since it is the most common centralised baseline method for comparison in the literature. For decentralised FL techniques, FedLEO and FELLO methods were selected owing to their close resemblance to our proposed method.

The experiments utilise 40 satellites that are sampled from the constellation. These satellites are clustered with different formations for MILP, and RL cluster formation approaches which are (11-11-11-7) and (5-7-3-3-5-7-3-7), respectively (numbers represent the size of each cluster). We repeat each experiment 3 times and use the average as the final result.

Datasets and Models: In order to conduct an extensive comparison between our proposed algorithm and the other approaches, we utilised diverse datasets, namely CIFAR-10, Imagenette and EuroSat. The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. The dataset is divided into five training batches and one test batch, each with 10000 images. Imagenette is a subset of 10 classified classes from the Imagenet dataset [52]. EuroSat [53] is a real satellite dataset, containing 27,000, 64×64pixel, 13-channel images collected by Sentinel-2A satellite, is used in this experiment. This dataset includes 10 classes, such as river, highway or forest. The RGB channels are utilised in our experiments. ResNet-18, ResNet-20 and VGG-11 are utilised as a training model for EuroSat, CIFAR10 and Imagenette datasets, respectively. Both IID and non-IID data distributions are considered among satellites. CIFAR-10 and Imagenette are considered in the IID distribution and EuroSat, which is a non-IID data dataset due to its spatial correlation and temporal variability characteristics, is used for the non-IID data distribution. We used both 32-bit single-precision (FP32) and 8-bit low-precision (BM8) formats in model training. For BM8, we have (4,3) and (2,5) representations.

TABLE IV
PERFORMANCE COMPARISON OF THE PROPOSED TECHNIQUES VS BASELINE FL TECHNIQUES IN FP32 FORMAT

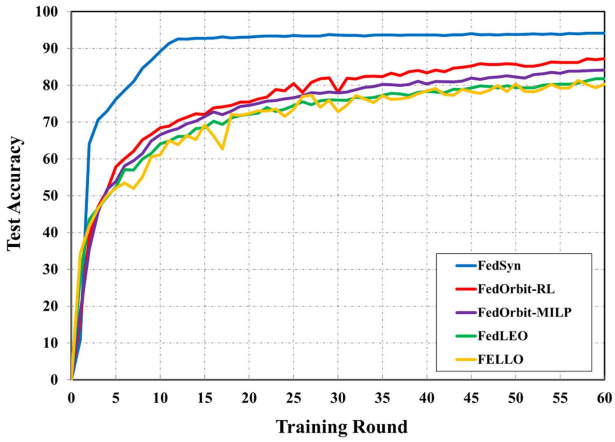
FL Approaches	Accuracy (%)			Training Time (Days)		
	CIFAR10	Imagenette	EuroSat	CIFAR10	Imagenette	EuroSat
FedSyn	94.10	73.53	96.06	30	20	30
FedOrbit-RL	87.24	66.27	89.86	3.35	2.85	3.05
FedOrbit-MILP	84.15	60.09	85.94	3.55	3.05	3.25
FedLEO	81.79	61.01	87.88	3.3	2.8	3
FELLO	80.35	59.83	85.73	1.6	1.35	1.45

B. Experimental Results

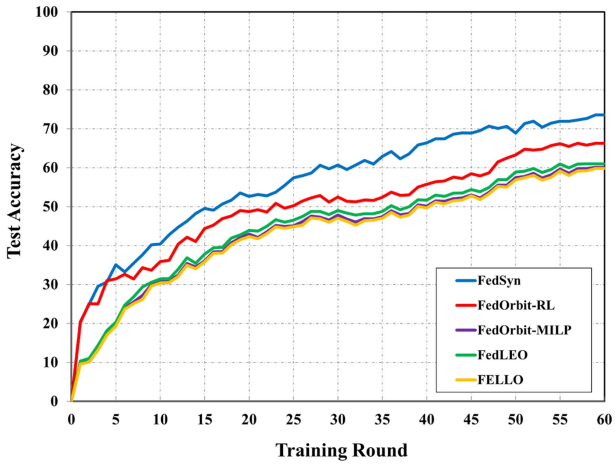
In this section, experimental results are discussed by considering three metrics, namely accuracy, training time and power.

1) Accuracy Analysis: Fig. 4 presents the training curve of the proposed FedOrbit as well as baseline techniques for all datasets and models by using FP32 numbering format. For all datasets and training models, the experiments report that the FedSyn (centralised FL) has the highest accuracy for CIFAR10, Imagenette and EuroSat datasets, which are 94.10%, 72.98%, and 96.36% respectively, due to more effective aggregation at the central server. The size of Imagnette dataset is the main reason for its lower accuracy than the others. Although in our setting, on average each satellite visits the GS 5 times during 24 hours, the interval between 2 consequent visits may be less than one round of training time. Based on the experiments, 2-3 training rounds can be completed per day for FedSyn method based on the model and dataset. So it takes almost 20-30 days for 60 training rounds as listed in Table IV. In addition to the long training time, communication with GS consumes more power than inter/intra satellite communications.

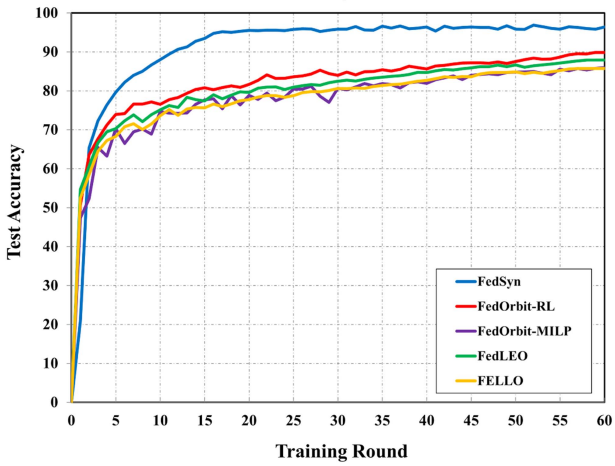
FedOrbit-RL (cluster formation via RL) outperforms FedOrbit-MILP (cluster formation via MILP), FedLEO and FELLO in terms of accuracy for all datasets and training models. On the other hand, although FedOrbit-MILP obtained slightly lower accuracy compared to FedLEO, its accuracy is comparable to FELLO. The results reveal the impact of cluster size on



(a) ResNet20-CIFAR10.



(b) VGG11-Imagenette.



(c) ResNet18-EuroSat

Fig. 4. Model accuracy of the proposed techniques vs baseline FL techniques (FP32 format).

accuracy, where FedOrbit-RL outperforms the others mostly due to the proper cluster number and size selection. FedOrbit-RL is better than FedOrbit-MILP around 3 to 6 percent for different models and datasets. Fig. 4(a) and (c) show that although the FELLO technique has higher accuracy at initial rounds, FedOrbit-MILP achieved higher accuracy at final rounds. This is related to the dataset size as well as the lower number of satellites in its cluster formation which leads to faster convergence and lower accuracy. The reason to utilise MILP was driven by its relatively simpler formulation compared to RL, which translates into faster problem-solving capabilities. This comparison represents a trade-off between computational efficiency and training accuracy. Specifically, the simplicity of MILP offers advantages in terms of reduced power consumption and lower computational overhead, though this comes with the drawback of reduced accuracy.

TABLE V
ACCURACY COMPARISON BETWEEN FP32 AND BM8

FL Approaches	Accuracy (%)		
	CIFAR10	Imagenette	EuroSat
FedSyn FP32	94.10	73.53	96.06
FedSyn BM(2,5)	93.62	73.28	95.61
FedSyn BM(4,3)	93.46	72.96	94.88
FedOrbit-RL FP32	87.24	66.27	89.86
FedOrbit-RL BM(2,5)	86.82	65.73	89.09
FedOrbit-RL BM(4,3)	86.62	65.15	88.23

It is worth noting that the accuracy differences in IID distribution are more significant that non-IID distribution. For instance, the accuracy difference between FedOrbit-RL and FedLEO for IID distribution (CIFAR10 & Imagenette datasets) is almost 5%, while this difference for non-IID distribution (EuroSat dataset) is less than 2%. This shows that although FL techniques may reach higher accuracy in IID distribution, accuracy improvement in non-IID distributions is more challenging. The impact of the number of contributed (clients) satellites in the FL procedure can be seen in Fig. 4(a), in which FELLO exhibits higher fluctuation due to considering fewer satellites in its operation.

Table V presents the impact of floating point precision on model accuracy on the proposed FedOrbit-RL and FedSyn for all datasets and models by using both FP32 and BM8 numbering formats. FedOrbit-RL is considered in this section due to its better performance compared to the others. Based on the obtained results, the training curve and convergence of BM8 was very similar to FP32 number format. From the accuracy point of view, the accuracy degradation in BM8 format is negligible compared with FP32. FedSyn FP32 achieved almost 0.51%, 0.34% and 0.46% higher accuracy than FedSyn BM(2,5) for CIFAR10, Imagenette and EuroSat datesets, respectively, as indicated in Table V.

In order to compare our proposed FedOrbit-RL BM8 with existing low precision approaches, Decentralised Low Precision Training (DLPT) approach [43] can be considered. Accuracy degradation for training VGG11 model on Imagenette dataset via DLPT approach is 2.19%, while it is only 0.54% for our proposed method. FedOrbit-RL FP32 achieved almost 0.48%, 0.81% and 0.88% higher accuracy than FedOrbit-RL BM(2,5) for CIFAR10, Imagenette and EuroSat datesets, respectively, as indicated in Table V. It demonstrates that, from an accuracy perspective, the influence of BM8 is more pronounced in decentralised methods compared to centralised methods. It is worth noting that although the accuracy difference among decentralised methods (i.e., FedOrbit, FedLEO and FELLO)

is significant, this difference is more considerable in IID data distribution as illustrated in Fig. 4. Hence, we can conclude that decentralised methods are more suitable for space applications due to the non-IID nature of space datasets.

2) Power Consumption and Training Time Analysis: The final accuracy and average training time of the proposed techniques and baseline FL techniques in FP32 numbering format for 60 training rounds are presented in Table IV. The best accuracy and worst training time are related to the FedSyn method due to its centralised nature which requires higher amount of communication with GS, aggregation in GS per round and waiting to receive models from all satellites.

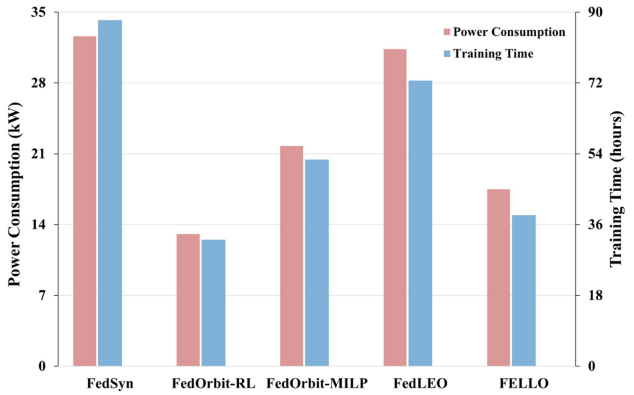
Based on experiments for both IID distributed datasets (i.e., CIFAR10 and Imagenette), and non-IID distributed dataset (EuroSat), the FedOrbit-RL method achieved higher accuracy (5.45%, 5.26%, and 1.98%, respectively) compared with the FedLEO method in almost the same time (including the waiting time for master satellites to enter the visibility zone of GS for final aggregation). Moreover, FedOrbit-RL outperforms FedOrbit-MILP in both accuracy and training time. This shows the impact of cluster size and formation on accuracy, where FedOrbit-RL reached higher accuracy due to using RL in its cluster determination procedure.

The effect of the number of sample satellites on accuracy and training time can be investigated by comparing FedOrbit-RL and FELLO methods, where FedOrbit-RL uses 40 satellites while FELLO uses 18 satellites in their training procedure. Although Fig. 4 shows that FedOrbit-RL has higher accuracy compared with FELLO method, Table IV shows that FELLO has the lowest training time among the selected approaches due to a lower number of communications as a consequence of the smaller number of sample satellites.

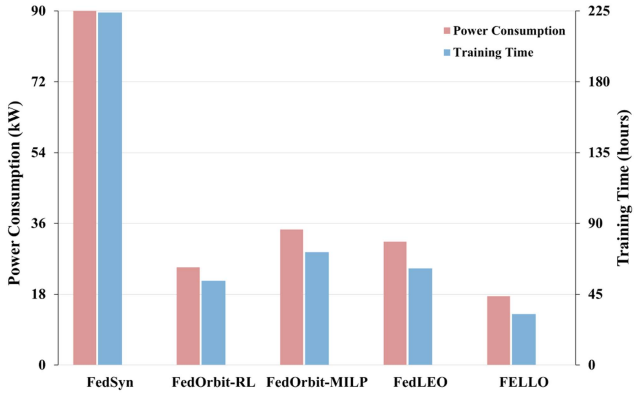
In addition, Table IV indicates that the decentralised methods are 7-10 times faster than FedSyn due to their decentralised structure, cluster formation, lower amount of communication with GS and master satellite selection. Moreover, it shows that model and dataset complexity and parameters have a direct impact on training time (ResNet-based models require more time than VGG-based approaches).

For a fair assessment of power consumption and training time, these metrics are evaluated at consistent accuracy levels across FL techniques. So, the baseline accuracy is set to the worst-performing method for a given dataset, defined as comparable accuracy (Comp. Acc.) in Fig. 5. The power consumption and training time of the ResNet20 model for CIFAR10 dataset for 80% accuracy is depicted in Fig. 5(a). FedOrbit-RL achieved a substantial speedup over FedSyn, FedOrbit-MILP, FedLEO and FELLO by 56 hours (63.4%), 20 hours (38.5%), 40 hours (55.5%), and 6 hours (15.8%), respectively. In addition, FedOrbit-RL achieved reductions of 39.9%, 58.3%, 25.3%, and 59.9% in power consumption compared to FedOrbit-MILP, FedLEO, FELLO, and FedSyn methods, respectively.

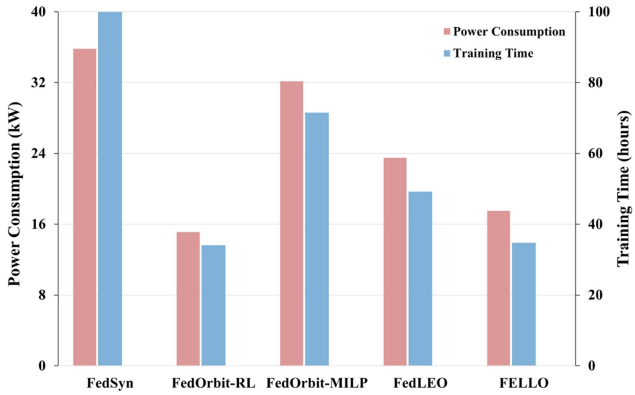
Fig. 5(b) depicts the power consumption and training time for the VGG11 model on the Imagenette dataset. The figure shows that FedOrbit-RL is 4 times faster than FedSyn due to its decentralised structure, prediction-based cluster formation and master satellite selection. Power consumption for FedOrbit-RL,



(a) ResNet20-CIFAR10 (Comp. Acc: 80%).



(b) VGG11-Imagenette (Comp. Acc: 60%).



(c) ResNet18-EuroSat (Comp. Acc: 85%).

Fig. 5. Training time and power consumption of the proposed techniques vs baseline FL techniques.

FedOrbit-MILP, FedLEO, FELLO, and FedSyn is 24.85, 34.45, 31.35, 17.5, and 90.07kW, respectively. This results in a 72.41% reduction by FedOrbit compared to FedSyn for achieving 60% accuracy. Similarly, Fig. 5(c) illustrates that FedOrbit performs better in terms of power consumption and training time of ResNet18 model for the EuroSat dataset, due to considering the GS visiting pattern as well as RL prediction in its master satellite selection.

3) Energy Optimisation Analysis: As mentioned before, BM8 is integrated with FL to reduce the satellites' energy

TABLE VI
COMPUTING TIME AND ENERGY CONSUMPTION FOR RESNET20 MODEL ON CIFAR10 DATASET

FL Approaches	Computing Time (h)		Avg. Energy Cons. (kWh)	
	Batch size 1	Batch size 32	Batch size 1	Batch size 32
FedSyn BM(2,5)	16.25	–	0.14137	–
FedSyn FP32 (CPU)	197.87	90.23	0.49468	0.22558
FedSyn FP32 (GPU)	85.00	5.19	5.95	0.36367
FedOrbit-RL FP32 (CPU)	–	33.50	–	0.08375
FedOrbit-RL FP32 (GPU)	–	1.93	–	0.13507

consumption. As 16 shows, consumed energy is calculated based on computing time and consumed power. Based on the experiments conducted by Guo et al. [11], the required computing time and consumed power for a batch size of 1, including the forward and backward paths for training the ResNet20 model on CIFAR10 dataset using BM(2,5) is 1.3ms and 8.7W, respectively. Given these results, we can estimate the computing time and energy consumption for FedSyn due to the similarity of the training model and process.

Based on our experimental results, the computing time for the ResNet20 model on the CIFAR10 dataset with a batch size of 1, and for FedSyn using FP32 on GPU and CPU are 6.8 and 15.83ms, respectively. We also obtained the same results with a batch size of 32 for both FedSyn and FedOrbit-RL techniques. In this case, the computing time for FedSyn using FP32 on GPU and CPU are 13.3 and 231ms, respectively. Moreover, the computing time for FedOrbit-RL using FP32 on GPU and CPU are 4.94 and 85.76ms, respectively. The number of the forward and backward paths for each epoch is calculated by dividing the number of data samples (60,000 for CIFAR10) by the batch size (1 or 32). The computing time for each epoch is calculated by 17. In addition, the consumed power on GPU and CPU are 70W and 2.5W, respectively.

Table VI shows the computing time and average energy consumption for the ResNet20 model on the CIFAR10 dataset for FP32 and BM(2,5) for 30 epochs and 25 training rounds. For the batch size of 1, BM(2,5) obtained significant improvement in both computing time and energy consumption compared with training on CPU and GPU. The results show that the utilisation of BM(2,5) as low-precision arithmetic enhanced the energy consumption by 3.5× compared with FP32 format on the CPU for the FedSyn method.

Although both CPU and GPU can handle batch size one, there are additional overheads associated with smaller batch sizes. This overhead includes the time it takes to load data onto the device and initialise computations. The experiments show that CPU (due to advanced vectorization units) and GPU (due to parallel processing) excel with larger batch sizes (i.e., 32), while, FPGA along with BM8 offers enhanced performance for low-latency, batch size one scenarios. In other words, Table VI shows that the energy consumption for BM8 with a batch size of one is more efficient than the CPU and GPU, and comparable with a batch size of 32.

VI. CONCLUSION AND FUTURE WORK

We proposed a novel decentralised federated learning approach, FedOrbit, designed specifically for Low Earth Orbit (LEO) satellite constellations. FedOrbit addresses the intricate challenges arising from intermittent connectivity among satellites, as well as the constraints posed by limited satellite power.

FedOrbit comprises a GS visiting pattern prediction algorithm, an RL-based cluster formation algorithm, and block minifloat arithmetic for model training to enhance training time and reduce power consumption. Our experimental results indicate that FedOrbit significantly improves training time and power consumption compared to both centralised and other decentralised FL techniques, with minimal impact on model accuracy. Furthermore, it achieves higher accuracy when compared to existing decentralised FL approaches for a range of datasets and machine learning models.

Future studies might focus on optimising cluster formation and inter-cluster communications/aggregations to further bolster model accuracy. Comparing existing cluster-based and quantisation-based approaches with our proposed BM-based FedOrbit-RL approach can be considered another future work. Additionally, validating our proposed power model through real-testbed implementation and hardware power measurements stands as a consideration for future studies.

REFERENCES

[1] T. Pfandzelter, J. Hasenburg, and D. Bermbach, "Towards a computing platform for the leo edge," in *Proc. 4th Int. Workshop Edge Syst., Analytics Netw.*, 2021, pp. 43–48.
[2] M. M. Gost, I. Leyva-Mayorga, A. Pérez-Neira, M. A. Vázquez, B. Soret, and M. Moretti, "Edge computing and communication for energy-efficient earth surveillance with leo satellites," in *Proc. IEEE Int. Conf. Commun. Workshops*, 2022, pp. 556–561.
[3] B. Denby and B. Lucia, "Orbital edge computing: Machine inference in space," *IEEE Comput. Archit. Lett.*, vol. 18, no. 1, pp. 59–62, May/Jun. 2019.
[4] C. Wu et al., "A comprehensive survey on orbital edge computing: Systems, applications, and algorithms," 2023, arXiv *2306.00275*.
[5] 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. Statist.*, 2017, pp. 1273–1282.
[6] A. U. Chaudhry and H. Yanikomeroglu, "Laser intersatellite links in a starlink constellation: A classification and analysis," *IEEE Veh. Technol. Mag.*, vol. 16, no. 2, pp. 48–56, Jun. 2021.
[7] C. Wu, Y. Zhu, and F. Wang, "Dsfl: Decentralized satellite federated learning for energy-aware leo constellation computing," in *Proc. 2022 IEEE Int. Conf. Satell. Comput.*, 2022, pp. 25–30.
[8] M. Elmahallawy and T. Luo, "Optimizing federated learning in leo satellite constellations via intra-plane model propagation and sink satellite scheduling," 2023, arXiv *2302.13447*.
[9] M. Elmahallawy and T. Luo, "Asyncfleo: Asynchronous federated learning for leo satellite constellations with high-altitude platforms," in *Proc. 2022 IEEE Int. Conf. Big Data*, 2022, pp. 5478–5487.
[10] C.-Y. Chen, L.-H. Shen, K.-T. Feng, L.-L. Yang, and J.-M. Wu, "Edge selection and clustering for federated learning in optical inter-leo satellite constellation," 2023, arXiv *2303.16071*.
[11] C. Guo, B. Lou, X. Liu, D. Boland, P. H. Leong, and C. Zhuo, "Boost: Block minifloat-based on-device CNN training accelerator with transfer learning," in *Proc. 2023 IEEE/ACM Int. Conf. on Comput. Aided Des.*, 2023, pp. 1–9.
[12] S. Fox et al., "A block minifloat representation for training deep neural networks," in *Proc. Int. Conf. Learn. Representations*, 2020.
[13] M. R. Jabbarpour, B. Javadi, P. H. Leong, R. N. Calheiros, D. Boland, and C. Butler, "On-board federated learning in orbital edge computing," in *Proc. 29th IEEE Int. Conf. Parallel Distrib. Syst.*, 2023, pp. 1045–1052.

[14] H. Chen, M. Xiao, and Z. Pang, "Satellite-based computing networks with federated learning," *IEEE Wirel. Commun.*, vol. 29, no. 1, pp. 78–84, Feb. 2022.
[15] B. Matthiesen, N. Razmi, I. Leyva-Mayorga, A. Dekorsy, and P. Popovski, "Federated learning in satellite constellations," *IEEE Netw.*, vol. 38, no. 2, pp. 232–239, Mar. 2024.
[16] D. Mäenpää, "Towards peer-to-peer federated learning: Algorithms and comparisons to centralized federated learning," Master's thesis, Linköpings University, Linköping, Sweden, 2021.
[17] N. Razmi, B. Matthiesen, A. Dekorsy, and P. Popovski, "Ground-assisted federated learning in leo satellite constellations," *IEEE Wireless Commun. Lett.*, vol. 11, no. 4, pp. 717–721, Apr. 2022.
[18] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," in *Proc. 12th Annu. Workshop Optim. Mach. Learn.*, 2020, pp. 1–11.
[19] N. Razmi, B. Matthiesen, A. Dekorsy, and P. Popovski, "On-board federated learning for dense leo constellations," in *Proc. IEEE Int. Conf. Commun.*, 2022, pp. 4715–4720.
[20] N. Razmi, B. Matthiesen, A. Dekorsy, and P. Popovski, "On-board federated learning for satellite clusters with inter-satellite links," 2023, arXiv *2307.08346*.
[21] N. Razmi, B. Matthiesen, A. Dekorsy, and P. Popovski, "Scheduling for ground-assisted federated learning in leo satellite constellations," in *Proc. 2022 30th Eur. Signal Process. Conf.*, 2022, pp. 1102–1106.
[22] M. Elmahallawy and T. Luo, "FedHAP: Fast federated learning for leo constellations using collaborative HAPs," in *Proc. 2022 14th Int. Conf. Wirel. Commun. Signal Process.*, 2022, pp. 888–893.
[23] J. So, K. Hsieh, B. Arzani, S. Noghabi, S. Avestimehr, and R. Chandra, "Fedspace: An efficient federated learning framework at satellites and ground stations," *arXiv preprint arXiv:2202.01267*, 2022.
[24] J. Lin, J. Xu, Y. Li, and Z. Xu, "Federated learning with dynamic aggregation based on connection density at satellites and ground stations," in *Proc. 2022 IEEE Int. Conf. Satell. Comput.*, 2022, pp. 31–36.
[25] L. Wu and J. Zhang, "Fedgsm: Efficient federated learning for leo constellations with gradient staleness mitigation," 2023, arXiv *2304.08537*.
[26] Z. Lin, Z. Chen, Z. Fang, X. Chen, X. Wang, and Y. Gao, "FeDSN: A general federated learning framework over leo satellite networks," 2023, arXiv *2311.01483*.
[27] M. Elmahallawy and T. Luo, "One-shot federated learning for leo constellations that reduces convergence time from days to 90 minutes," 2023, arXiv *2305.12316*.
[28] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.
[29] H. H. Yang, Z. Liu, T. Q. Quek, and H. V. Poor, "Scheduling policies for federated learning in wireless networks," *IEEE Trans. on Commun.*, vol. 68, no. 1, pp. 317–333, Jan. 2020.
[30] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2016, arXiv *1610.05492*.
[31] C. Hardy, E. Le Merrer, and B. Sericola, "Distributed deep learning on edge-devices: Feasibility via adaptive compression," in *Proc. 2017 IEEE 16th Int. Symp. Netw. Comput. Appl.*, 2017, pp. 1–8.
[32] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-IID data," *IEEE Trans. Neural Netw. Syst.*, vol. 31, no. 9, pp. 3400–3413, Jan. 2020.
[33] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1709–1720.
[34] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "FedPAQ: A communication-efficient federated learning method with periodic averaging and quantization," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2020, pp. 2021–2031.
[35] S. Horváth, D. Kovalev, K. Mishchenko, P. Richtárik, and S. Stich, "Stochastic distributed learning with gradient quantization and double-variance reduction," *Optim. Methods Softw.*, vol. 38, no. 1, pp. 91–106, 2023.
[36] N. Shlezinger, M. Chen, Y. C. Eldar, H. V. Poor, and S. Cui, "UVeQFed: Universal vector quantization for federated learning," *IEEE Trans. Signal Process.*, vol. 69, pp. 500–514, 2020.
[37] Y. Li, W. Li, and Z. Xue, "Federated learning with stochastic quantization," *Int. J. Intell. Syst.*, vol. 37, no. 12, pp. 11600–11621, 2022.
[38] R. Hönig, Y. Zhao, and R. Mullins, "Dadaquant: Doubly-adaptive quantization for communication-efficient federated learning," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 8852–8866.
[39] X. Dai et al., "Hyper-sphere quantization: Communication-efficient SGD for federated learning," 2019, arXiv *1911.04655*.
[40] S. Chen, C. Shen, L. Zhang, and Y. Tang, "Dynamic aggregation for heterogeneous quantization in federated learning," *IEEE Trans. Wireless Commun.*, vol. 20, no. 10, pp. 6804–6819, Oct. 2021.
[41] Y. Liu, P. Huang, F. Yang, K. Huang, and L. Shu, "QuAsyncFL: Asynchronous federated learning with quantization for cloud-edge-terminal collaboration enabled AIoT," *IEEE Internet Things J.*, vol. 11, no. 1, pp. 59–69, Jan. 2024.
[42] Y. Ji and L. Chen, "FedQNN: A computation–communication-efficient federated learning framework for IoT with low-bitwidth neural network quantization," *IEEE Internet Things J.*, vol. 10, no. 3, pp. 2494–2507, Feb. 2023.
[43] S. A. Aketi, S. Kodge, and K. Roy, "Low precision decentralized distributed training over IID and non-IID data," *Neural Netw.*, vol. 155, pp. 451–460, 2022.
[44] H. Kaushal and G. Kaddoum, "Optical communication in space: Challenges and mitigation techniques," *IEEE Commun. Surveys Tut.*, vol. 19, no. 1, pp. 57–96, First Quarter 2017.
[45] V. W. Chan, "Free-space optical communications," *J. Lightw. Technol.*, vol. 24, no. 12, pp. 4750–4762, 2006.
[46] IEEE Standard for Floating-Point Arithmetic, IEEE Std 754–2019 (Revision of IEEE 754-2008), 2019.
[47] Y. Jing et al., "Energy-efficient routing based on a genetic algorithm for satellite laser communication," *Opt. Express*, vol. 31, no. 5, pp. 8682–8695, 2023.
[48] Y. Wang, J. Zhang, X. Zhang, P. Wang, and L. Liu, "A computation offloading strategy in satellite terrestrial networks with double edge computing," in *Proc. 2018 IEEE Int. Conf. Commun. Syst.*, 2018, pp. 450–455.
[49] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tut.*, vol. 19, no. 4, pp. 2322–2358, Fourth Quarter 2017.
[50] I. Ali, N. Al-Dhahir, and J. E. Hershey, "Predicting the visibility of LEO satellites," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 35, no. 4, pp. 1183–1190, Oct. 1999.
[51] C. He et al., "FedMLl: A research library and benchmark for federated machine learning," 2020, arXiv *2007.13518*.
[52] J. Howard et al., "Imagenette," 2020. [Online]. Available: https://github.com/fastai/imagenette
[53] P. Helber, B. Bischke, A. Dengel, and D. Borth, "EuroSAT: A novel dataset and deep learning benchmark for land use and land cover classification," *IEEE J. Sel. Top. Appl. Earth Observ. Remote Sens.*, vol. 12, no. 7, pp. 2217–2226, Jul. 2019.

Mohammad Reza Jabbarpour is currently a research fellow with Western Sydney University, Australia. Prior to this appointment, he was assistant professor with the ICT department of Niroo Research Institute (NIR), Tehran, Iran. He also served as an assistant professor with the Department of Computer Engineering, Islamic Azad University North Tehran Branch, Iran, from 2016 to 2018. He has received various awards including WiC 2015: Asia Invention Association Grand Award, Gold medal, PECIPTA 2015: Bronze Medal, Singapore Challenge: Merit award for his patented idea. His current research interests include blockchain, satellite resiliency, vehicular network and connected vehicles, federated learning, and machine learning.

Bahman Javadi (Senior Member, IEEE) is a full professor in networking and cloud computing with Western Sydney University, Australia. He has received multiple national awards including the IoT Impact Awards and InnovationAUS Awards for Excellence for his research projects. His research interests include cloud computing, Edge computing, performance evaluation of large-scale distributed computing systems, and reliability and fault tolerance. He is a senior member of ACM, executive committee member of the *IEEE Technical Committee on Cloud Computing* (TCCLD), and senior fellow of the Higher Education Academy of U.K.

Philip H.W. Leong received the BSc, BE and PhD degrees from the University of Sydney. In 1993 he was a consultant to ST Microelectronics in Milan, Italy working on advanced flash memory-based integrated circuit design. From 1997 to 2009 he was with the Chinese University of Hong Kong. He is currently professor in computer systems in the School of Electrical and Computer Engineering, University of Sydney, visiting professor with Imperial College, chief technology advisor with ClusterTech and chief technology officer with CruxML.

Rodrigo N. Calheiros is an associate professor with the School of Computer, Data and Mathematical Sciences, Western Sydney University, Australia. He has worked in the field of Cloud computing and related areas since 2008, and since them he carried out R&D supporting research in the area. His research interests also include Big Data, Internet of Things, fog computing, and their application.

David Boland received the MEng degree (Hons.) in information systems engineering and the PhD degree from Imperial College London, London, U.K., in 2007 and 2012, respectively. From 2013 to 2016, he was with Monash University, Melbourne, VIC, Australia, as a lecturer before moving to The University of Sydney, Sydney, NSW, Australia. His current research interests include numerical analysis, optimisation, design automation, and machine learning.