2023 24th IEEE International Conference on Mobile Data Management (MDM) | 979-8-3503-4101-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/MDM58254.2023.00020

One-Shot Federated Learning for LEO Constellations that Reduces Convergence Time from Days to 90 Minutes

Mohamed Elmahallawy
Department of Computer Science
Missouri University of Science and Technology
Rolla, MO 65409, USA
meqxk@mst.edu

Tie Luo*
Department of Computer Science
Missouri University of Science and Technology
Rolla, MO 65409, USA
tluo@mst.edu

***Abstract*—A Low Earth orbit (LEO) satellite constellation consists of a large number of small satellites traveling in space with high mobility and collecting vast amounts of mobility data such as cloud movement for weather forecast, large herds of animals migrating across geo-regions, spreading of forest fires, and aircraft tracking. Machine learning can be utilized to analyze these mobility data to address global challenges, and Federated Learning (FL) is a promising approach because it eliminates the need for transmitting raw data and hence is both bandwidth and privacy-friendly. However, FL requires many communication rounds between clients (satellites) and the parameter server ($\mathcal{PS}$), leading to substantial delays of up to several days in LEO constellations. In this paper, we propose a novel *one-shot* FL approach for LEO satellites, called *LEOShot*, that needs only *a single communication round* to complete the entire learning process. LEOShot comprises three processes: (i) *synthetic data generation*, (ii) *knowledge distillation*, and (iii) *virtual model retraining*. We evaluate and benchmark LEOShot against the state of the art and the results show that it drastically expedites FL convergence by *more than an order of magnitude*. Also surprisingly, despite the one-shot nature, its model accuracy is on par with or even outperforms regular iterative FL schemes by a large margin.**

***Index Terms*—Satellite communications, low Earth orbit (LEO), federated learning, knowledge distillation, ensemble model, synthetic data generation, teacher-student framework.**

I. Introduction

Low Earth orbit (LEO) satellite constellations have recently been burgeoning due to the rapid advances in satellite communications (SatCom) technology. Positioned at an altitude of 160–2,000 km above the Earth's surface, LEO satellites are often equipped with sensors and high-resolution cameras to collect a vast amount of mobility-related data, such as tracking and monitoring cloud movements for weather forecast [1], hurricane and forest fire movements [2], flooding situations, migration of large herds of animals across geographic regions, and aircraft tracking. Large-scale machine learning (ML) models can be utilized to analyze these mobility data to address global challenges such as climate change, natural disasters, and abnormal wildlife conditions. However, traditional ML methods require downloading raw data such as satellite images to a ground station (GS) or gateway for centralized model training, which is not practical for SatCom because of its limited bandwidth, large propagation delay, and privacy concerns (e.g., satellite data and images may contain sensitive information such as military activities or critical infrastructure locations).

Introducing federated learning (FL) [3] to SatCom appears to be a viable solution because FL eliminates the need for transmitting raw data by allowing satellites to train ML models locally (i.e., on-board) using their own data respectively and only send the resulting model parameters to the GS which acts as the $\mathcal{PS}$ to aggregate those local models into a global model. However, FL requires many communication rounds between clients (satellites) and the $\mathcal{PS}$ to re-train and re-aggregate the models until it converges into a well-functioning global model. As a result, this iterative process can take several days or even weeks in the context of SatCom [4], [5], because of the long propagation delay and, most importantly, the highly *sporadic and irregular connectivity* between LEO satellites and the GS. The latter is attributed to the distinct trajectories of satellites and the GS,[1] which leads to very intermittent and non-cyclic visits of satellites to the GS (successively).

In this paper, we propose LEOShot, a novel *one-shot* FL approach for LEO satellite constellations that accomplishes the FL training process in a single communication round, yet still obtaining a global model with competitive performance. One-shot FL is an emerging paradigm [6] but its existing methods cannot be directly applied to SatCom because they (i) require the $\mathcal{PS}$ to have a publicly shareable dataset that represents the client data distribution [7], [8], which is hardly available, (ii) require clients to upload raw or

*Corresponding author. This work is supported by the National Science Foundation under Grant No. 2008878.

[1]A satellite orbits at an *inclination angle* between 0^o and 90^o (typically 50-80^o), whereas a GS constantly rotates on the 0^o plane. These degrees are in reference to the Equator of the Earth.

2375-0324/23/$31.00 ©2023 IEEE
DOI 10.1109/MDM58254.2023.00020

synthetic data [9], [10], which contradicts the FL principle, or (iii) still needs extra communication rounds to achieve a satisfactory accuracy [11]. In contrast, LEOShot does not share or transmit data in any form (e.g., raw or synthetic) yet still attains a high-performing model in just a single communication round. In summary, this paper makes the following contributions:

- To the best of our knowledge, LEOShot is the first one-shot FL approach proposed for SatCom. It drastically reduces the adverse effect of highly sporadic and irregular connectivity between satellites and GS by instating only one communication round. Not only this, but it also operates without relying on any auxiliary datasets or raw-data sharing, yet attaining competitive model performance.
- LEOShot comprises (i) *synthetic data generation* that generates images mimicking real satellite images instead of downloading them; (ii) *knowledge distillation* that trains (instead of aggregates) a global model using a teacher-student framework; iii) *virtual model retraining* that refines the global model iteratively toward high performance but without any extra communication rounds.
- Unlike conventional FL which assumes an identical ML model architecture for all clients, LEOShot allows clients to use *different* model architectures to cater to their own preferences and resource constraints. This is much more flexible and helps solve the *data heterogeneity* and *system heterogeneity* among LEO satellites.
- We demonstrate via extensive simulations that LEOShot dramatically accelerates FL convergence by *more than an order of magnitude* as compared to the state-of-the-art FL-SatCom approaches. In the meantime, the accuracy of its trained models outperforms existing methods by a large margin despite its one-shot nature.

Paper Organization. Section II provides an overview of recent work that utilizes the FL in SatCom. Section III describes the FL-SatCom network model as well as communication links among satellites and GS. Section IV explains the proposed LEOShot framework and its component in detail. The performance evaluation of LEOShot is provided in Section V. Finally, Section VI concludes this paper.

II. Related Work

A. FL for SatCom

While FL-SatCom is still in its infancy, a few studies have attempted to apply FL to SatCom. Chen et al. [12] directly applied FedAvg [13] to SatCom to demonstrate FL's advantages in avoiding the need to download raw data to a GS. FedISL [14] leverages intra-plane inter-satellite-link (ISL) to reduce the long waiting time for satellites to become visible to the $\mathcal{PS}$, but only performs well under an ideal setup where the $\mathcal{PS}$ is either a GS located at the North Pole (NP) or a medium Earth orbit (MEO) satellite positioned directly above the Equator. In addition, FedISL overlooks the *Doppler shift* resulting from the high relative speed between LEO satellites (clients) and MEO-satellite ($\mathcal{PS}$). Another approach called FedHAP [15] was proposed which introduces one or multiple high altitude platforms (HAPs) floating at 18-25km above the Earth's surface as $\mathcal{PS}$. As a result of using HAPs, the convergence speed of FL is improved by having more visible satellites, but more hardware is required. Another most recent approach, FedLEO [16], has been proposed to enhance the convergence process of FL in the LEO constellation. FedLEO achieved this through the use of intra-plane model propagation and scheduling of sink satellites. It is however necessary for each satellite to run a scheduler to determine which satellite will be the sink to send its model, resulting in a delay for models to be exchanged with the GS.

The above are all *synchronous FL* approaches. There are also *asynchronous FL* approaches proposed for SatCom that allow the $\mathcal{PS}$ to proceed to the next training rounds without waiting for the model updates from *all* the clients. Razmi et al. [17] proposed FedSat, which assumes that the GS is located at NP to simplify the problem so that every satellite visits the GS at *regular intervals* (once every orbital period). To overcome this limitation, they proposed another approach FedSatSchedule [5], which uses a scheduler to reduce model staleness by predicting satellites' visiting patterns to the GS. However, several days are required to reach a model convergence. Another recent approach called FedSpace [4] formulated an optimization problem to dynamically schedule model aggregation based on predicting connectivity between LEO satellites and the GS, but it requires each satellite to upload a small portion of its data so that the GS can schedule model aggregation, which contradicts FL's principle of avoiding raw data sharing. Another recently proposed FL-SatCom approach, AsyncFLEO [18], is proposed to offer a drastic solution to the staleness challenge that requires several days for asynchronous FL-SatCom approaches to converge. It is capable of achieving convergence within a few hours, but is still subject to a high number of communication rounds and hence there is still large room for improvement.

B. One-Shot FL

To the best of our knowledge, one-shot FL has not been introduced into SatCom before. The studies [7], [8] employ knowledge distillation [19] in which client-side models are used as teacher models to train a server-side student model (global model). However, the server is required to have access to a public dataset in order to provide pseudo samples for training purposes, which conflicts with FL's privacy principles. As an alternative to knowledge distillation, data distillation [20] is used in [9], [10]. These studies, however, show a notable underperformance; moreover, they require uploading distilled/synthetic data generated by clients to a

$\mathcal{PS}$, which is incompatible with FL principles. Lastly, a theoretical analysis of one-shot FL is presented in [21], focusing on independently and identically distributed (IID) data only.

Given the above challenges, none of the existing one-shot frameworks can be practically applied without issues or drawbacks. In addition, to develop a one-shot FL approach for SatCom, it is necessary to take into account the SatCom challenges such as long propagation delays, intermittent and irregular visibility of LEO satellites, and bandwidth limitations.

III. Network Model

Fig. 1 illustrates an LEO constellation $\mathcal{M}$ consists of N satellites equally distributed on O orbits. Each orbit $o \in \mathcal{O} = \{o_1, o_2, ..., o_O\}$ is located at an altitude h_o above the Earth with an inclination angle α_o and has a set of satellites $\mathcal{M}_o$. Satellites in an orbit o travel with the same velocity v_o and has the same orbital period T_o. Here, $v_o = \sqrt{\frac{GM_E}{(R_E+h_o)}}$ and $T_o = \frac{2\pi}{\sqrt{GM_E}}(R_E+h_o)^{3/2}$, where G is the gravitational constant, M_E is the Earth's mass, and $R_E = 6,371$ km is the Earth's radius.

In SatCom, satellite m can communicate with a GS g if the Earth does not obstruct their line-of-sight (LoS) link. Mathematically, this means $\vartheta_{m,g}(t) \triangleq \angle(r_g(t), (r_m(t) - r_g(t))) \leq \frac{\pi}{2} - \vartheta_{min}$, where $r_m(t)$ and $r_g(t)$ are the trajectories of m and g, respectively, and ϑ_{min} is the minimum elevation angle that depends on the GS location.

A. FL-SatCom System

Consider an FL-SatCom system, in which each satellite $m \in \mathcal{M}$ collects a set of Earth images $\mathcal{D}_m$ of size $d_m = |\mathcal{D}_m|$. In addition, we assume that these images are non-IID among satellites since they orbit the Earth at irregular intervals and scan different areas. In a synchronous FL system such as FedAvg [13], the $\mathcal{PS}$ (i.e., a GS) and satellites train an ML model collaboratively to solve the following problem

$$\arg\min_{w \in \mathbb{R}^d} F(w) = \sum_{m \in \mathcal{M}} \frac{d_m}{d} F_m(w), \tag{1}$$

where $F(w)$ indicates the overall loss function (e.g., SSE) of the target model; w is the model weights; $d = \sum_{m \in \mathcal{M}} d_m$ is the total data size; $F_m(w)$ is the loss function of satellite m, which can be expressed as

$$F_m(w) = \frac{1}{d_m} \sum_{j=1}^{d_m} f_m(w; x_{m,j}), \tag{2}$$

where $f_m(\cdot)$ is the training loss on a sample point $x_{m,j}$.

During the training process, there are multiple communication rounds $\beta = \{0, 1, 2, \dots\}$. In each round, the GS first transmits the latest global model weights w^β to all satellites, and each satellite m employs a local optimization scheme such as gradient descent to update its model for I epochs as

$$w_m^{\beta,i+1} = w_m^{\beta,i} - \eta \nabla F_m(w_m^{\beta,i}; x_m^i), \; i = 0, 1, 2, ..., I-1 \tag{3}$$

where η is the learning rate. Following that, each satellite uploads its locally updated model to the GS for assembling as

$$w^{\beta+1} = \sum_{m \in \mathcal{M}} \frac{d_m}{d} w_m^{\beta,I}, \tag{4}$$

which completes a communication round. The above procedure iterates with an incremental β until the FL model converges (e.g., a target accuracy, target loss, or the maximum number of training rounds is reached).

A key challenge of this learning process arises from the fact that the convergence of FL requires several communication rounds between LEO satellites and the GS, and each communication can only happen when a satellite transiently comes into the GS's visible zone. As a result, FL would take several days or even longer to reach convergence. This motivated us to develop LEOShot, which requires only a single communication round between satellites and the GS.

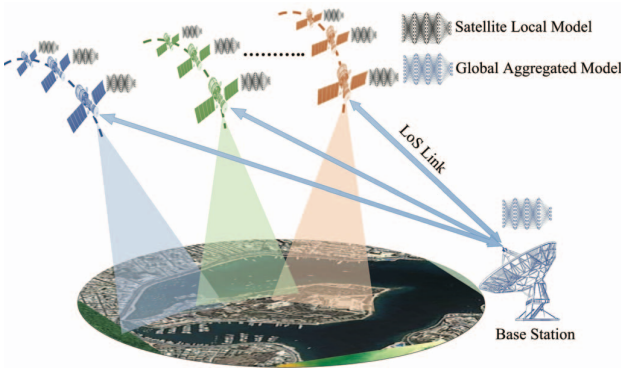


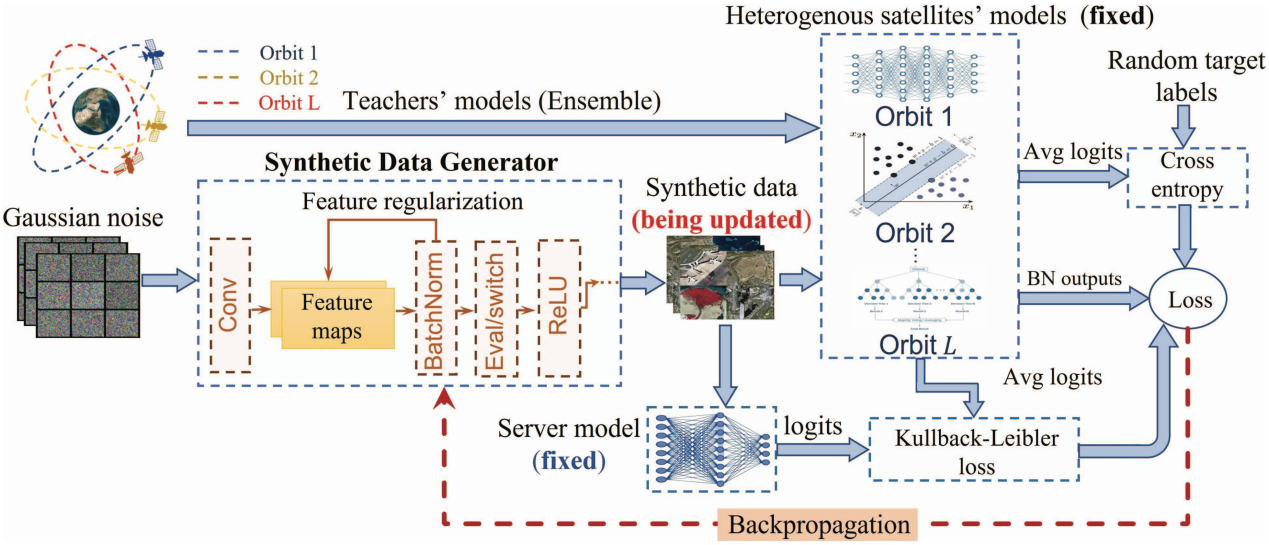
Fig. 1: FL-LEO network architecture, comprised of multiple orbits each having multiple satellites.

B. Communication Model

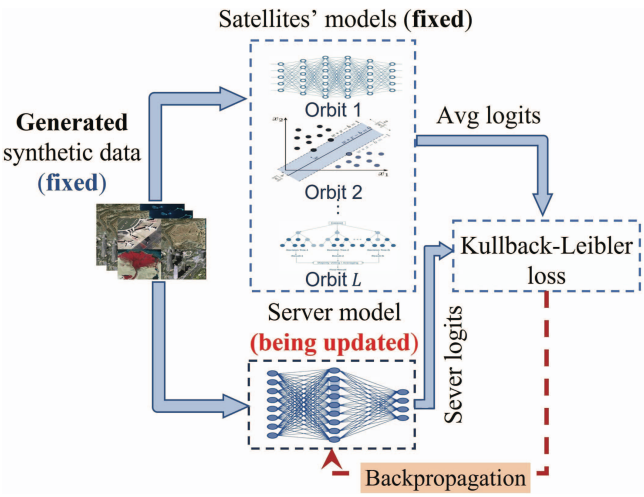
In a symmetric radio frequency channel with additive white Gaussian noise (AWGN), the signal-to-noise ratio (SNR) between a satellite m and a GS g can be expressed as [22]:

$$SNR(m, g) = \frac{PG_mG_g}{KTBL_{m,g}}, \tag{5}$$

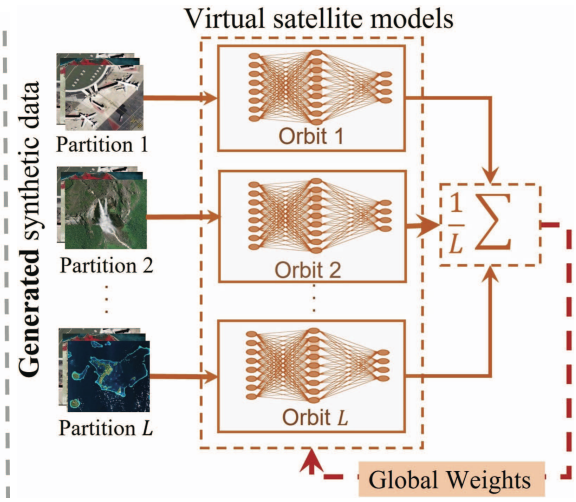
where P is the transmitter power, G_m and G_g are the total antenna gains of satellite m and an GS g, respectively, K is the Boltzmann constant ($1.38 \times 10^-23 J/K$), T is the noise temperature at the receiver, B is the channel bandwidth, and $L_{m,g}$ is the free-space pass-loss between satellite m and a



(a) **Phase 1:** Synthetic data generation.



(b) **Phase 2:** Knowledge distillation.



(c) **Phase 3:** Virtual model retraining.

Fig. 2: Overview of the proposed LEOShot framework.

GS g. As long as the LoS link between a satellite m and a GS g is not obstructed by the Earth, then $L_{m,g}$ can be expressed as

$$L_{m,g} = \left(\frac{4\pi \|m, g\|_2 f}{c}\right)^2 \tag{6}$$

where $\|m, g\|_2$ is the Euclidean distance between a satellite m and a GS g when they are visible to each other, f is the carrier frequency, and c is the light speed. For exchanging local or global model weights (w_m or w) between a satellite m and a GS g, the total required time t_t can be calculated as

$$t_t = \underbrace{\frac{z|\mathcal{P}|}{R}}_{\textbf{Transmission delay}} + \underbrace{\frac{\|m, g\|_2}{c}}_{\textbf{Propagation delay}} + t_m + t_s \tag{7}$$

where t_m and t_s are the processing delay at *m-th* satellite and a GS g, respectively, $|\mathcal{P}|$ is the number of sample points, z is the number of bits in each sample, and R is the maximal achievable data rate, which can be computed by the Shannon

formula as

$$R \approx B \log_2(1 + SNR(m, g)) \tag{8}$$

IV. The LEOShot framework

LEOShot is operated at the server end. Fig. 2 provides an overview of the LEOShot framework, which is comprised of three phases: (a) synthetic data generation, which synthesizes a representative dataset as a *proxy* of all the satellites' local data, (b) knowledge distillation, for distilling information from the satellite models (teacher models) to train a server model (student model), and (c) virtual model retraining, for training virtual local models iteratively until the server model converges. Algorithm 1 outlines the entire process of LEOShot.

Algorithm 1: LEOShot's 3-phase process

Input: Satellites' models, L, η_g, η_s, x, y, J,I ,γ_1, γ_2
Output: Server model parameters
▷ **Phase 1:** `Synthetic data generation`
Generate batches of random noises x and labels y
Initialize server model (student) and generator $\mathcal{G}$
foreach *epoch* j *of training* $\mathcal{G}$ **do**
  Calculate losses $\mathcal{R}_{CE}, \mathcal{R}_{BN}, R_{KL_{Gen}}(\hat{x})$ using eqns. (10), (11), and (12)
  Calculate the generator loss $\mathcal{R}'_{Gen}$ via (14)
  Generate/Update $\hat{x}$
  Retrain the weights of $\mathcal{G}$ on generated $\hat{x}$ via (14)
end
▷ **Phase 2:** `Knowledge distillation`
foreach *epoch* j *of training server model* **do**
  Calculate loss $\mathcal{R}_{KL_S}$ of the server model via (16)
  Retrain the weights of the server model via (17)
end
return w_s
▷ **Phase 3:** `Virtual Model Retraining`
Generate L virtual models
Cluster $\hat{x}$ into L Partitions
foreach *epoch* j *of updating server model* **do**
  foreach *virtual model* w_l **do**
    ▷ `All models train in parallel`
    Initialize $w_l \leftarrow w_s$
    Train w_l using its assigned partition of $\hat{x}$
  end
  Update $w_s \leftarrow \frac{1}{L} \sum_{l=1}^{L} w_l$
end
return w_s

As a background, the server begins its operation once it receives all the client models from all the orbits via *sink satellites*. A sink satellite [15] is a satellite on each orbit who (1) collects all the models from other satellites on the same orbit (via intra-orbit model relay), (2) assembles them together into a *partial global model* [15] and (3) sends it to the server.

A. Synthetic Data Generation

The objective of this phase is to build a generator $\mathcal{G}$ to generate high-quality unlabeled synthetic data (i.e., having sufficient features for correct classification) without having to download any real data from satellites to a GS or requiring any auxiliary (i.e., publicly available) datasets. To achieve this objective, we use the ensemble of the client (satellite) models uploaded by various sink satellites to generate synthetic data (see Fig. 2a). Note that a salient feature of LEOShot is that it allows *heterogeneous model architectures* of these client models. That is, each client can decide and use its own preferred neural network rather than a common architecture used by all the satellites (as dictated by the standard FL). We achieve this by using an *ensemble* of all the client models and *train* the target global model via *knowledge distillation* (instead of by averaging model weights).

Our generator $\mathcal{G}$ was inspired by the generative adversarial network (GAN) [23], but is distinct from it as we do not require public datasets. Given a randomly generated input x (e.g., Gaussian white noise) and a random label y, our generator attempts to generate synthesized data $\hat{x}$ with similar features to the data collected by satellites, by solving the following optimization problem:

$$\min_{\hat{x}} \mathcal{R}_{Gen} \triangleq \mathcal{R}_{CE}(D(\hat{x}), y) + \mathcal{R}(\hat{x}), \tag{9}$$

where $\mathcal{R}_{Gen}$ denotes the overall loss of the generator, $\mathcal{R}_{CE}(\cdot)$ is a cross-entropy loss (e.g., classification loss), and $\mathcal{R}(\hat{x})$ is a regularization term used to steer $\hat{x}$ towards more realistic data (e.g., images). Here, $D(\hat{x})$ is the average logits of the ensemble model given an input $\hat{x}$ (where logits are the output of usually the last fully connected layer), and is defined by

$$D(\hat{x}) = \frac{1}{|\mathcal{M}|} \sum_{l \in \mathcal{L}} f_l(\hat{x}, w_{\mathcal{K}_l}), \tag{10}$$

where $f_l(\hat{x}, w_{\mathcal{K}_l})$ is an estimation function of the ensemble model received from orbit l, that returns the logits of $\hat{x}$ when the model parameter $w_{\mathcal{K}_l}$ is given. Eq. (10) allows us to (indirectly) measure how well the generated synthetic data $\hat{x}$ mimics both the distribution and the particular instances of the original satellite data, without accessing the original data. In fact, attempting to access satellites' training data contradicts the FL principles. Moreover, unlike traditional FL algorithms (e.g., FedAvg), our design of generator (9) uses logits (as in $D(\cdot)$) instead of client model weights, which enables our approach to deal with *heterogeneous* client models.

The regularizer $\mathcal{R}(\hat{x})$ serves the purpose of improving the stability of the generator. The need comes from the fact

that the ensemble of satellite models is trained on non-IID datasets, and therefore tend to make the generator unstable and get stuck in suboptimal local minima. Additionally, there is a risk of overfitting the synthetic data, which ultimately hinders the generator from achieving high levels of accuracy [24]. Therefore, we use a BatchNorm (BN) layer during training to normalize the feature maps to reduce the impact of covariate shifts [25] and overcome the gradient vanishing problem (so that the distance between synthetic images and original satellite images can be continuously reduced). In addition, this BN layer also implicitly stores the average logits as channel-wise means and variances. Thus finally, the $\mathcal{R}(\hat{x})$ realized by this BN layer can be expressed by

$$\mathcal{R}_{BN}(\hat{x}) = \frac{1}{L}\sum_{l\in\mathcal{L}}\sum_{b}\Big(\|\mu_b(\hat{x}) - \mu_{l,b}\|_2 + \|\sigma_b^2(\hat{x}) - \sigma_{l,b}^2\|_2\Big) \quad (11)$$

where $\mu_b(\hat{x})$ and $\sigma_b^2(\hat{x})$ are the batch-wise estimation of the mean and variance, respectively, associated with the *b-th* BN layer of the generator, and $\mu_{l,b}$ and $\sigma_{l,b}^2$ are the mean and variance of the *b-th* BN layer of $f_l(\hat{x}, w_{\mathcal{K}_l})$. As a result of adding this regularization term, our designed generator outputs synthetic images with high quality that are close to the original satellite images.

B. Knowledge Distillation

In this phase, we strive to distill the knowledge from the ensemble of all the client models to train a server (i.e., global) model. Although the generated synthetic data has high quality, it is not useful for knowledge distillation because of the large gap between the ensemble model's decision boundaries and the server model's decision boundary [26]. To address this issue, we force the generator to generate more synthetic data with different distributions and then employ Kullback-Leibler (KL) divergence to minimize the distance between the predictions (proxy of decision boundary) of the ensemble model and the server model during synthetic data generation (bottom of Fig. 2a). The new regularizer term that represents the KL divergence is

$$\mathcal{R}_{KL_{Gen}}(\hat{x}) = 1 - \tfrac{1}{2}\big\{KL(D_G(\hat{x}), Q) + KL(D_E(\hat{x}), Q))\big\} \quad (12)$$

$$Q = \frac{1}{2}\cdot(D_G(\hat{x}) + D_E(\hat{x})), \quad (13)$$

where $KL(\cdot)$ denotes the KL divergence loss, $D_G(\hat{x})$ is the server model's logits, and $D_E(\hat{x})$ is the ensemble model's average logits.

Thus, with our BN and KL regularization terms, we reformulate our generator optimization problem (9) as

$$\mathcal{R}'_{Gen} = \min_{\hat{x}}\Big[\mathcal{R}_{CE}(D(\hat{x}), y) + \gamma_1\mathcal{R}_{BN}(\hat{x}) + \gamma_2 R_{KL_{Gen}}(\hat{x})\Big] \quad (14)$$

where γ_1 and γ_2 are hyper-parameters to trade-off between the two losses. In addition, we optimize the weights of our generator $\mathcal{G}$ for J epochs using SGD as follows:

$$w_{Gen}^j = w_{Gen}^{j-1} - \eta_g\nabla\mathcal{R}_{Gen}(w_{Gen}^{j-1}; (\hat{x}, y)^{j-1}), \quad j = 1, 2, \ldots, J \quad (15)$$

where w_{Gen}^j is the generator's model at iteration j, and η_g is the $\mathcal{G}$'s learning rate.

After these optimization procedures, our generator can generate synthetic images not only of high quality but also of a distribution that resembles the original satellite data to enable effective knowledge distillation.

Referring to Fig. 2b, the next step after generating the synthetic data $\hat{x}$ is to commence the updating and retraining of the server model until it attains an acceptable accuracy. To this end, the synthetic data $\hat{x}$ is fed to both the ensemble of satellite models which acts as the teacher, and the server model which acts as the student. Subsequently, the average logits are computed as the outcome of training the teacher model using (10), which can be used for both homogeneous and heterogeneous ensemble models. The average logits are then applied to distill the knowledge from the ensemble (teacher) model to the server (student) model by minimizing the prediction error between the two models, through the KL divergence as follows:

$$\mathcal{R}_{KL_S}(\hat{x}) = KL(D(\hat{x}), D_s(\hat{x})) \quad (16)$$

where $D_s(\hat{x})$ is the logits of the server model after being trained on the generated synthetic data $\hat{x}$. Since the satellite models are trained on non-IID data, we aim to improve the accuracy of the server model and address the issue of poor performance or divergence problem encountered in [24]. To achieve this, we further optimize the server model parameters by employing SGD as

$$w_s^j = w_s^{j-1} - \eta_s\nabla\mathcal{R}_{KL_S}(w_s^{j-1}; \hat{x}^{j-1}), \quad j = 1, 2, \ldots, J \quad (17)$$

where w_s^j is the server model at iteration j, and η_s denotes the learning rate of the server model. Note that our method is different from [24] which uses local batch normalization at each client to harmonize local data distributions but requires several rounds of communication between server and clients. Instead, we use generated synthetic data to resemble the original satellite data and it allows us to directly train a server model locally using SGD, without the need for communication or averaging satellite models which are influenced by non-IID data distributions. On the downside, synthetic data may not be as good as real data; to address this, in Phase 3 we retrain our server model to improve model accuracy.

As a result of this phase, we have successfully trained a server model that leverages the knowledge from the ensemble satellite models and the generated synthetic data, and we have taken into account the possible heterogeneity of both the data and models involved. Fig. 2b illustrates the entire knowledge distillation process.

C. Virtual Model Retraining

Although the knowledge distillation phase outputs a functional server model, the model performance still has notable

room for improvement (e.g., the classification accuracy was merely 70% on the MNIST dataset). Certainly, allowing for extra communication rounds (and aggregation) between the GS and the LEO satellites will improve model accuracy, but this clearly contradicts our goal of one-shot learning and will also negatively impact the convergence speed. Thus, we propose a novel method that transforms the distributed FL into a *localized* version, by creating *virtual local models* on the server and trains those models *locally* until the server model converges. Specifically, our method consists of four steps: (i) clone L copies of the server model to serve as the initial virtual local models, (ii) partition the generated synthetic data (after labeling them using a K-means clustering algorithm) into L groups, each with the same class distribution as one of the L orbits (distributions were received at the end of model dissemination), (iii) train each virtual model on one of the L data groups, (iv) aggregate the weights of these trained virtual models to obtain an updated server model. The above repeats until the server model converges, which is the final global model. Fig. 2c illustrates the entire retraining process for virtual models.

V. Performance Evaluation

A. Simulation setup

LEO Constellation. We consider a Walker-delta constellation $\mathcal{M}$, which consists of 40 LEO satellites distributed over five orbits, each with eight satellites. Each orbit is located at an altitude h_o of 500km above the Earth's surface with an inclination angle of 80°. A GS is located in Rolla, MO, USA (can be anywhere) with a minimum elevation angle ϑ_{min} of 10°. For both LEOShot and baselines, Table I (upper part) summarizes the parameters pertaining to the communication links described in Section III-B. By using Systems Tool Kit (STK), a software tool for analyzing satellite constellations, we extract the visibility between satellites and the GS. To obtain each set of results, we simulate communication between satellites and the GS over a period of three days.

TABLE I: Simulation Parameters (upper: communication; lower: training)

Parameters	Values
Transmission power (satellite & GS) P	40 dBm
Antenna gain of (satellite & GS) G_m, G_s	6.98 dBi
Carrier frequency f	2.4 GHz
Noise temperature T	354.81 K
Transmission data rate R	16 Mb/sec
Number of local training epochs I	300
Learning rate η	0.001
Mini-batch size b_k	32
Generator learning rate η_g	0.001
Weighting factors γ_1&γ_2	1 & 10

Baselines. We compare LEOShot with the state-of-the-art approaches that were proposed most recently and are reviewed in Section II, including FedSpace [4], FedISL [14], FedHAP [15], FedSat [17], and FedSatSchedule [5].

ML models and Dataset. An important implication of LEOShot's capability of allowing heterogeneous client models (cf. Section IV-A), is that the server no longer needs to broadcast an initial model w^0 to all the clients, which in the context of SatCom will save a significant amount of time. However, for comparison with existing methods, we assume a standard (homogeneous) FL setting where the neural network architecture is common and broadcasting w^0 is still required. *We highlight that this setup substantially favors baseline approaches* and not ours. In the experiment, all satellites train a ResNet-50 model. For each baseline, the GS aggregates the satellite models into a global ResNet-50 model. For LEOShot, however, since a key component of knowledge distillation is to train a smaller global model, the GS trains a ResNet-18 model.

For comparison purposes, we use the same datasets as all the baseline approaches use: MNIST [27] and CIFAR-10 [28]. Additionally, we consider a non-IID setting for both datasets, where satellites in two orbits are trained with 4 classes while satellites in the other three orbits are trained with the remaining 6 classes. The lower part of Table I summarizes the training hyperparameters.

B. Results

Comparison with Baselines. As shown in Fig. 3, LEOShot achieves the fastest convergence (on both datasets) in only **90 minutes** with an accuracy of 85.64% on MNIST attained in a single communication round (the convergence time includes waiting for at least one satellite per orbit to be visible to upload a partial model to the GS). The second fastest approach is FedISL [14] with the *ideal* setup described in Section II (GS at NP or MEO above the Equator), which takes 3.5 hours for FL to converge with an accuracy of 82.67%. Without the ideal setup, its convergence time spikes to 72 hours, and accuracy drops to 61.19%. FedSat [17] and FedHAP [15] have marginally higher accuracy than LEOShot but their convergence is significantly slower. Also importantly, FedSat assumes an ideal setup (GS at NP) similar to FedISL, and FedHAP requires extra hardware (HAP) of substantial extra cost. FedSatSchedule [5] does not have FedSat's ideal assumption, and as a result, its accuracy is only 76.32% and its convergence time doubles FedSat.

For all methods, accuracy on CIFAR-10 is lower than on MNIST after the same amount of training time, which is particularly prominent for FedSat [17]. On the other hand, LEOShot maintains a very small difference, which demon-

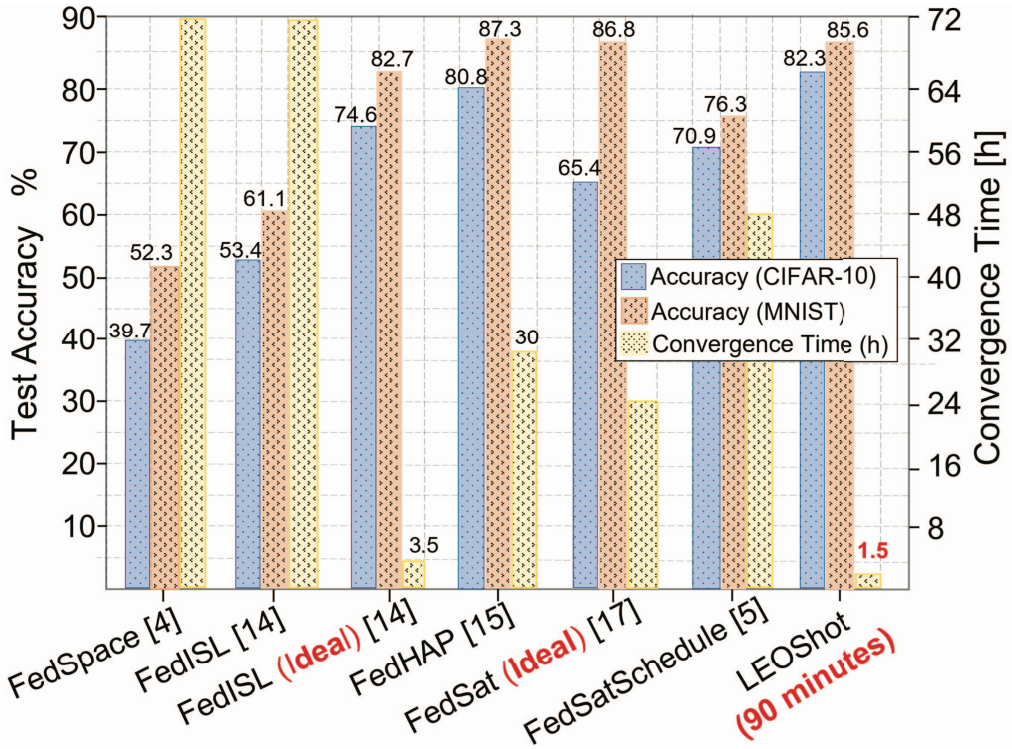


Fig. 3: Accuracy and Convergence Time comparison on non-IID data. For all approaches, convergence time was first measured on MNIST and then fixed for CIFAR-10 to measure the accuracy.

strates certain robustness. Recall that LEOShot achieves high accuracy in just a *single communication round*.

Effectiveness of using synthetic data in knowledge distillation. Pertaining to Section IV-A and IV-B, this subsection (i) investigates whether using model *logits* or model *parameters* (weights or gradients) is more effective in producing a server model, and (ii) assesses the test accuracy on multiple deep learning (DL) models with the generated synthetic data. Fig. 4 shows a sample of the generated synthetic images using logits from satellites trained on MNIST and CIFAR-10 datasets. These images approximate the distribution and mimic the content of the real images very well, enabling an effective knowledge distillation.

TABLE II: Comparison of DL models trained on synthetic images

DL model	Accuracy (%)	
	MNIST	CIFAR-10
CNN (2 layers)	62.26	60.88
VGG-11	69.15	62.72
Wide-ResNet-40-1 [29]	71.51	66.79
ResNet-18	**73.64**	**70.67**

For the first purpose, we conducted two experiments on LEOShot. In the first experiment, satellite model parameters are uploaded to the GS similarly to traditional FL. In the second experiment, model logits are uploaded instead. All the satellites train a ResNet-50 model on MNIST in the non-IID setting. Fig. 5 shows the resulting test accuracy of each partial global model derived from each orbit, as well as the accuracy after averaging the weights or logits. The results indicate that the test accuracy of each partial global model varies depending on the data imbalance from each orbit. The accuracy of the server model is only 31.74% when averaging these partial models' parameters, which underperforms the individual partial models. In contrast, when averaging the logits, the server model achieves an accuracy of 62.36%, which outperforms all the individual partial models. This interesting finding confirms that a single communication round is far from sufficient for weight averaging (as in traditional FL) to accommodate the discrepancy between client models trained on non-IID data; but on the other hand, averaging logits allows knowledge distillation through our *local* training that minimizes the distance between logits of our teacher and student models. In addition, using logits also allows us to accommodate model heterogeneity.

(a) Samples of generated synthetic images (MNIST dataset).

(b) Samples of generated synthetic images (CIFAR-10 dataset).

Fig. 4: Samples of synthetic images created by our generator when satellites trained on various datasets.

For the second purpose, Table II reports the performance of various DL models when trained using our synthesized images. We can see that all the DL models achieve acceptable accuracy, with ResNet-18 achieving the highest 73.64% on MNIST. This set of results validates that the quality of our synthesized images is reliable, which plays an important role in transferring knowledge to the server model.

Impact of virtual model retraining. Pertaining to Section IV-C, here we investigate how our use of virtual local models affects the accuracy of the server model. As reported in Table II, the server model achieves an accuracy of 73.64% by using ResNet-18 as the target model and MNIST as the training dataset. Note that this is achieved without virtual model retraining. Now, with that, we clone five virtual ResNet-18 models initialized with the initial

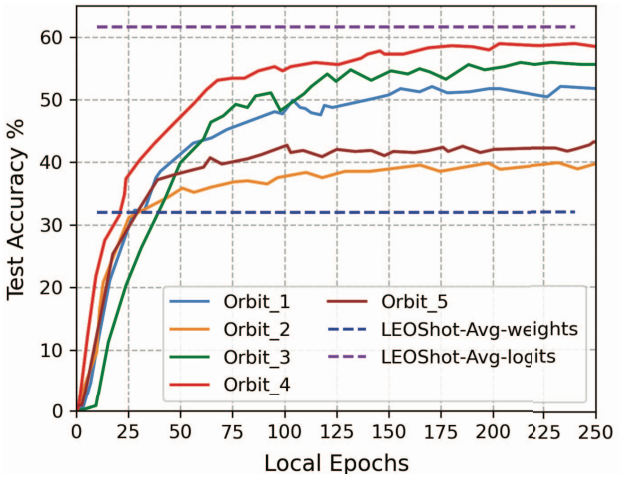


Fig. 5: Test accuracy of partial models from sink satellites on different orbits (solid lines) and the best accuracy obtained by averaging logits or weights (dotted lines).

server model weights and train each virtual model on one of five partitions of synthetic images. The final server model is obtained via weight averaging. Our result in Fig. 3 shows that the server model accuracy increases to 85.64% which is a significant 16% improvement without requiring any extra communication round.

VI. Conclusion and Future Work

This work makes the first effort to introduce one-shot FL into SatCom. We propose a novel framework called LEOShot to address the challenge of highly sporadic and irregular visits of LEO satellites to a GS. Unlike prior work, LEOShot does not require public datasets or client data uploads, thus upholding important FL principles on data privacy protection and communication efficiency. In addition, unlike standard FL which dictates a common identical neural network architecture for all the clients and the server, LEOShot allows each client to choose its own preferred ML model based on its computing resources and data properties. In our quantitative study in comparison with the state-of-the-art benchmarks, we find that LEOShot reduces FL training/convergence time drastically up to 80 times (it converges in as short as 90 minutes); in the meantime, it achieves high accuracy even under challenging non-IID settings and outperforms the benchmarks by large margins.

In our future work, we aim to examine LEOShot on real and diverse satellite datasets in different settings. This would include exploring a variety of LEO constellations ranging from sparse to dense constellations with GS located at different geographical locations, as well as training heterogeneous ML models across satellites and constellations.

REFERENCES

[1] A. Perez-Portero, J. F. Munoz-Martin, H. Park, and A. Camps, "Airborne gnss-r: A key enabling technology for environmental monitoring," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 14, pp. 6652–6661, 2021.

[2] P. Barmpoutis, P. Papaioannou, K. Dimitropoulos, and N. Grammalidis, "A review on early forest fire detection systems using optical remote sensing," *Sensors*, vol. 20, no. 22, p. 6442, 2020.

[3] P. Kairouz *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[4] J. So *et al.*, "FedSpace: An efficient federated learning framework at satellites and ground stations," *arXiv:2202.01267*, 2022.

[5] N. Razmi, B. Matthiesen, A. Dekorsy, and P. Popovski, "Scheduling for ground-assisted federated learning in LEO satellite constellations," in *2022 30th European Signal Processing Conference (EUSIPCO)*, 2022, pp. 1102–1106.

[6] S. Salehkaleybar, A. Sharifnassab, and S. J. Golestani, "One-shot federated learning: theoretical limits and algorithms to achieve them," *The Journal of Machine Learning Research*, vol. 22, no. 1, pp. 8485–8531, 2021.

[7] N. Guha, A. Talwalkar, and V. Smith, "One-shot federated learning," *arXiv:1902.11175*, 2019.

[8] Q. Li, B. He, and D. Song, "Practical one-shot federated learning for cross-silo setting," *arXiv:2010.01017*, 2020.

[9] A. Kasturi, A. R. Ellore, and C. Hota, "Fusion learning: A one shot federated learning," in *International Conference on Computational Science*. Springer, 2020, pp. 424–436.

[10] Y. Zhou, G. Pu, X. Ma, X. Li, and D. Wu, "Distilled one-shot federated learning," *arXiv:2009.07999*, 2020.

[11] J. Zhang, C. Chen, B. Li, L. Lyu, S. Wu, J. Xu, S. Ding, and C. Wu, "A practical data-free approach to one-shot federated learning with heterogeneity," *arXiv:2112.12371*, 2021.

[12] H. Chen, M. Xiao, and Z. Pang, "Satellite-based computing networks with federated learning," *IEEE Wireless Communications*, vol. 29, no. 1, pp. 78–84, 2022.

[13] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[14] N. Razmi, B. Matthiesen, A. Dekorsy, and P. Popovski, "On-board federated learning for dense LEO constellations," in *IEEE International Conference on Communications (ICC)*, May 2022.

[15] M. Elmahallawy and T. Luo, "FedHAP: Fast federated learning for LEO constellations using collaborative HAPs," in *2022 14th International Conference on Wireless Communications and Signal Processing (WCSP)*. IEEE, 2022.

[16] M. Elmahallawy and T. Luo, "Optimizing federated learning in LEO satellite constellations via intra-plane model propagation and sink satellite scheduling," in *ICC 2023-IEEE International Conference on Communications*, 2023. [Online]. Available: {https://arxiv.org/abs/2302.13447}

[17] N. Razmi, B. Matthiesen, A. Dekorsy, and P.Popovski, "Ground-assisted federated learning in LEO satellite constellations," *IEEE Wireless Communications Letters*, 2022.

[18] M. Elmahallawy and T. Luo, "AsyncFLEO: Asynchronous federated learning for LEO satellite constellations with high-altitude platforms," in *2022 IEEE International Conference on Big Data (BigData)*, 2022, pp. 5478–5487.

[19] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *International Journal of Computer Vision*, vol. 129, no. 6, pp. 1789–1819, 2021.

[20] T. Wang, J.-Y. Zhu, A. Torralba, and A. A. Efros, "Dataset distillation," *arXiv:1811.10959*, 2018.

[21] S. Salehkaleybar, A. Sharif-Nassab, and S. J. Golestani, "One-shot federated learning: Theoretical limits and algorithms to achieve them." *J. Mach. Learn. Res.*, vol. 22, pp. 189–1, 2021.

[22] I. Leyva-Mayorga, B. Soret, and P. Popovski, "Inter-plane inter-satellite connectivity in dense LEO constellations," *IEEE Trans. on Wireless Communications*, vol. 20, no. 6, pp. 3430–3443, 2021.

[23] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, "Ensemble distillation for robust model fusion in federated learning," *Advances in Neural Information Processing Systems*, pp. 2351–63, 2020.

[24] X. Li, M. Jiang, X. Zhang, M. Kamp, and Q. Dou, "Fedbn: Federated learning on non-iid features via local batch normalization," *arXiv preprint arXiv:2102.07623*, 2021.

[25] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.

[26] B. Heo, M. Lee, S. Yun, and J. Y. Choi, "Knowledge distillation with adversarial samples supporting decision boundary," in *Proceedings of AAAI*, 2019, pp. 3771–3778.

[27] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[28] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (canadian institute for advanced research)," *URL http://www. cs. toronto. edu/kriz/cifar. html*, vol. 5, no. 4, p. 1.

[29] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv:1605.07146*, 2016.