SpaceDML: Enabling Distributed Machine Learning in Space Information Networks

Hanxi Guo, Qing Yang, Hao Wang, Yang Hua, Tao Song, Ruhui Ma, and Haibing Guan

***Abstract*—Space Information Networks (SINs) has become a rapidly growing global infrastructure service. Massive volumes of high-resolution images and videos captured by low-orbit satellites and unmanned aerial vehicles (UAVs) have provided a rich training data source for machine learning applications. However, SIN devices' limited communication and computation resources make it challenging to perform machine learning efficiently with a swarm of SIN devices.**

In this paper, we propose SpaceDML, a distributed machine learning system for SIN platforms that applies dynamic model compression techniques to adapt distributed machine learning training to SINs' limited bandwidth and unstable connectivity. SpaceDML has two key algorithms: 1) Adaptive loss-aware quantization that compresses models without sacrificing their quality. 2) Partial weight averaging that selectively averages active clients' partial model updates. These algorithms jointly improve communication efficiency and enhance the scalability of distributed machine learning with SIN devices. We evaluate SpaceDML by training a LeNet-5 model on the MNIST dataset. The experimental results show that SpaceDML can increase model accuracy by 2-3% and reduce communication bandwidth consumption by up to 60% compared with the baseline algorithm.

***Index Terms*—machine learning; model compression; space information networks**

I. Introduction

The last decade has witnessed a rapid development of space information networks (SINs) and platforms such as small satellites and unmanned aerial vehicles (UAVs) for public services, including Internet service, geographical photography, navigation, weather forecasting, and traffic data analysis. So far, SpaceX has already deployed over 1,000 StarLink satellites to provide broadband Internet connectivity. Besides, small satellites and UAVs equipped with sensors and cameras have been collecting tons of high-resolution imagery and video data, which enable applications such as Google Maps to provide real-time street views and traffic monitoring.

The increasing volumes of valuable data collected by SIN devices have inspired a broad spectrum of interests for intelligent and efficient data analysis. Leveraging the data to enable machine intelligence on satellites and UAVs has two significant benefits: *First*, improving data analysis efficiency and further enhancing the quality of public services, such as weather forecast, transportation navigation. *Second*, assisting spacecraft management and autonomy to optimize space communications and spacecraft reliability, further reducing the burden and cost of the ground segment and mission operations.

Motivated by these benefits, both industry and academy have attempted to enable machine learning on SIN devices. KP Labs has developed *Leopard*, an onboard computer integrated with a powerful FPGA chip that accelerates the execution of deep learning algorithms on satellites [1]. Intel released the Movidius Myriad VPU, a microprocessor capable of accelerating machine vision tasks in a low-powered environment such as satellites and UAVs. Besides, European Space Agency explored machine learning techniques to optimize space communications that—*Mexar2*—is proposed to determine the best timing of transmitted data packets to improve downlink capability [2]. NASA has designed the Space Communications and Navigation (SCaN) Testbed to leverage machine learning to explore cognitive radio and underused portions of the electromagnetic spectrum for communication.

However, existing studies only attempted to perform machine learning on a single SIN device, even though machine learning applications can be extensively accelerated by parallel computing. For example, Manning *et al*. [3] deployed the machine learning framework TensorFlow Lite to a modern small satellite computer and performed image classification tasks using convolutional neural networks (CNNs). Nevertheless, it is impossible to obtain accurate and general machine learning models on a single SIN device due to its limited computation capacity and storage space. Existing deep learning algorithms typically take millions of data samples and thousands of parallel computing threads (*i.e.*, GPU streaming processors) to train a model qualified for production environments. Therefore, it becomes imperative to enable distributed machine learning (DML) with a swarm of satellites and UAVs.

The obstacle that hinders deploying machine learning to a swarm of SIN devices comes from the scarce frequency and orbit resources that limit the communication efficiency between SIN devices and ground operators. There is a contradiction between the huge traffic demands of machine learning applications and SIN platforms' communication quality. Moreover, the unstable connectivity of SINs further increases the difficulty to orchestrate machine learning among a swarm of satellites and UVAs.

In this work, we propose SpaceDML, a distributed machine learning (DML) framework for SIN platforms. SpaceDML deploys a global server that sends out machine learning

Hao Wang is the corresponding author.

H. Guo, Q. Yang, T. Song, R. Ma and H. Guan are with the Shanghai Key Laboratory of Scalable Computing and Systems, School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China (Email: {hanxiguo, yangqinghush, songt333, ruhuima, hbguan}@sjtu.edu.cn)

H. Wang is with the Computer Science & Engineering, Louisiana State University, Baton Rouge, LA 70803-2804, USA (Email: haowang@lsu.edu)

Y. Hua is with the EEECS/ECIT, Queen's University Belfast, Belfast BT7 1NN, UK (Email: Y.Hua@qub.ac.uk)

tasks to each SIN device and iteratively updates a global model by exchanging model weights with SIN devices—following a new distributed machine learning paradigm—federated learning [4]. With multiple SIN devices participating in training, SpaceDML tolerates unstable SIN connectivities where disconnected devices either fail or extensively slow down the training. Besides, SpaceDML applies the latest model compression algorithm that reduces the size of model weights to accommodate the limited bandwidth of SINs and allows more SIN devices to participate in distributed training. We propose to perform adaptive loss-aware model quantization that compresses local models trained on SIN devices without sacrificing model accuracy. We also develop a quantized weight averaging algorithm to aggregate quantized model weights from each SIN device efficiently. The main contributions of SpaceDML are as follows:

- We design SpaceDML, an efficient distributed machine learning framework that extensively reduces the communication overheads among SIN devices.
- We introduce an adaptive loss-aware model compression method into distributed machine learning with SIN devices that adapts to the limited bandwidth and unstable connectivity in SIN.
- We evaluate SpaceDML using a realistic model and a public dataset. Experimental results of training a LeNet-5 model on the MNIST dataset show that SpaceDML can increase accuracy by 2-3% and reduce communication overheads between SIN devices by up to 60% compared with the baseline algorithm FedAvg.

II. Preliminaries

This section first reviews existing SIN services and then introduces the background of federated learning and model compression.

A. Space Information Network Services

Space Information Networks (SINs) are global network infrastructures carried by space devices distributed at different attitudes through integrated network interconnecting, including satellites, airships, aerostats, and UAVs. Using such a complex hierarchical architecture, SINs bring connectivity across the world and enable broadband Internet access services to users anywhere and anytime. A wide range of industrial and academic applications have benefited from SINs' global coverage, especially remote sensing, artificial intelligence, and public transportation [3, 5, 2].

Integrating machine intelligence to SINs has become practical as more and more SIN devices are equipped with high-performance computing chips and precise sensors. Instead of collecting data from SIN devices to terrestrial stations, directly training machine learning models on SIN devices mitigates the long data transmission time between SIN devices and terrestrial stations thanks to the instant access to massive volumes of valuable data captured by SIN devices [3]. However, the huge network traffic generated by distributed machine learning is challenging for existing SIN infrastructures. Existing studies on distributed machine learning have not specifically optimized for the limited and unstable communication conditions in SINs.

B. Federated Learning

Federated Learning is a new distributed machine learning paradigm that orchestrates a large scale of devices to perform on-device training and global model update [4]. Unlike traditional machine learning algorithms that centralize data to a server for training, each participating device k in federated learning trains a model w_k locally and exchanges the local model w_k with a server to update a global model w. As centralizing data from SIN devices is time-consuming, federated learning can exploit the immediate on-device data access to avoid data transmission latency. As SIN devices capture new data, federated learning can update models locally without waiting for data centralization as traditional distributed machine learning (DML) does.

However, due to its distributed nature, federated learning still needs to consume network bandwidth to exchange model weights between clients and a centralized server. A few studies have been proposed to further minimize the communication overhead in federated learning by compressing the model weights exchanged between devices. Konečný *et al.* [6] designed structured updates and sketched updates to reduce data transmission. LotteryFL [7] applies the Lottery Ticket hypothesis with FedAvg, applying model pruning algorithms to remove less important model weights without sacrificing model accuracy. FedSketchedSGD [8] compresses gradients using the Count Sketch algorithm to decrease communication bandwidth usage.

C. Model Quantization

Deep neural networks typically have a large number of layers and channels that result in a significant redundancy. There are three popular model compression techniques: model pruning, knowledge distillation, and model quantization. Model pruning may spoil the integrity of the whole model structure and lose some contributing neurons in layers. Knowledge distillation requires the teacher networks and student networks to be rigorously harmonious. Otherwise, the learning process may be unstable. Model quantization makes it possible to deploy deep neural networks to devices with limited computation resources and storage by keeping every model parameter but approximates floating-point numbers by lower bit-width numbers. BinaryConnect [9] mapped floating-point parameters to +1 or −1 using either deterministic or stochastic operation. Binary-encoded floating-point numbers can exploit efficient bit-wise operation in microprocessors to accelerate computation. Courbariaux *et al.* [10] proposed quantizing model parameters and activations by mapping the intermediate product of input to a lower bit. ALQ [11] designed a method to train a multi-bit network based on the loss function instead of the reconstruction error, allowing that the parameters in the same layer may have different precision after model quantization.

A few model quantization techniques have been proposed to improve the communication efficiency in federated learning,

reducing the volume of updates transmitted between the global server and participating devices. Quantized SGD [12] that quantizes gradient and trades off between communication bandwidth and converging times in distributed setting was proposed to boost efficiency. FedPAQ [13] combined period averaging, partial participation, and quantization with federated learning to address communication bottleneck. Nir Shlezinger *et al.* [14] proved coupling universal quantization vector with Federated Learning works well. LFL algorithm [15] quantized client models and global server models so that the transmission can be further reduced. However, all these existing works follow a static model quantization policy that can hardly adjust model compression rate dynamically.

III. SpaceDML's Design

This section first describes SpaceDML's architecture and deployment in SINs. We then introduce the two key parts of SpaceDML: 1) Adaptive loss-aware quantization (ALQ) for multi-bit neural networks; 2) Partial averaging and ALQ for global model weights. At last, we present a detailed complexity analysis of the algorithm applied in SpaceDML.

A. Architecture and Deployment in SINs

Fig. 1 presents the architecture of SpaceDML, where there are K SIN devices participating in training and one SIN device working as the global server. The global server device orchestrates participating devices to train local models and exchange model weights iteratively. The participating SIN devices communicate with the global server device using SIN communication channels.

In each communication round, the server device picks a subset of K devices in SINs and distributes its global model weights to them as in Algorithm 1. When the selected devices receive the quantized global weights, they reconstruct the original global weights with small precision loss. Then, these devices use the reconstructed weights to train the model using their local data. After the training, each selected device uses ALQ to quantize the local weights and uploads the quantized local weights $\boldsymbol{B}_i$ and $\boldsymbol{\alpha}_i$ to the server device. For any participating device i, $\boldsymbol{B}_i$ is the concatenated binary bases of all layer weight groups, and $\boldsymbol{\alpha}_i$ is the corresponding vectorized coordinates. The server device reconstructs and aggregates all the local weights to update the global model. And the server uses ALQ to quantize the global model. After all participating devices receive the updated quantized global model, a new communication round begins.

B. Problem Formulation

In this paper, we aim to formulate a distributed machine learning paradigm for SINs, with an objective to reduce the volume of data transmitted between SIN devices. In SpaceDML, we formulate the distributed machine learning procedure into the following optimization problem:

$$\min_{\boldsymbol{B},\boldsymbol{\alpha}} f(\boldsymbol{B},\boldsymbol{\alpha};X) = \frac{1}{K}\sum_{i=1}^{K} f_i(\boldsymbol{B}_i,\boldsymbol{\alpha}_i;X_i), \tag{1}$$

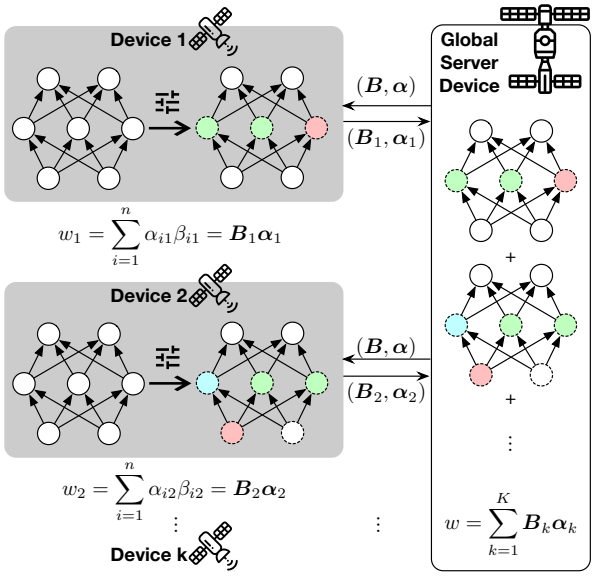

Fig. 1: An overview of SpaceDML's architecture.

where f_i is the loss function on a device i, K is the number of participating SIN devices, and X_i is the input training data on the device i. $\boldsymbol{B}$ is the concatenated binary bases and $\boldsymbol{\alpha}$ is the corresponding vectorized coordinates. Combining $\boldsymbol{B}$ and $\boldsymbol{\alpha}$ can reconstruct the model weights, which will be introduced in Section III-C.

The objective of SpaceDML is to reduce the communication overhead introduced by model weight exchange between SIN devices and allow more devices to participate in training. Quantizing model weights into lower-precision data formats can extensively reduce communication bandwidth usage. With the same level of infrastructures and bandwidth provisioning, reducing the communication overhead of each device will allow more devices to participate in training, eventually increasing the training efficiency.

C. Adaptive Loss-aware Model Compression

Quantization reduces communication overhead by converting higher-precision model parameters to lower-precision ones. Preliminary quantization maps each model parameter to a lower-precision model parameter, while multi-bit networks quantize their weights to coordinates and binary bases, further reducing the data volume required to keep the model. However, ALQ can generate multi-bit networks with several bit-widths in the same layer, compared with global bit-width in one layer for common multi-bit networks. The ALQ algorithm keeps the composition of the model and reduces the model size extensively.

Adaptive loss-aware quantization (ALQ) for multi-bit networks divides the same layers into disjoint groups and quantizes them into diverse bit-widths [11]. For weight $w_l \in \mathbb{R}^{N\times 1}$ in layer l of the model on device i, $N = n \times M$. ALQ separates the weights of layer l into M groups. Each group $\hat{w}^i_{l,m}$, $m \in M$ has its own $\boldsymbol{B}^i_{l,m}$ and $\boldsymbol{\alpha}^i_{l,m}$.

$$\hat{w}^i_{l,m} = \sum_{j=1}^{I^i_{l,m}} \alpha_j\beta_j = \boldsymbol{B}^i_{l,m}\boldsymbol{\alpha}^i_{l,m}, \tag{2}$$

where $\beta_j \in \{-1,+1\}^{n\times 1}$, $\boldsymbol{B}^i_{l,m} \in \{-1,+1\}^{n\times I^i_{l,m}}$, $\alpha \in \mathbb{R}_+^{I^i_{l,m}\times 1}$, and $I^i_{l,m}$ is the bit-width.

We apply the ALQ workflow to quantize local models on each device. A local model is trained through pre-training, ALQ, and post-training with the local data. The pre-training period consists of full-precision training offering the prerequisite for ALQ quantization. It should be noted that the pre-training should only be operated in the first round for the devices participating in global communication.

The ALQ algorithm has three phases. The first phase is pruning the model in the α domain, which is the coordinates for binary bases. The binary bases $\boldsymbol{B}$ will have to accord change. Here, pruning is an approximation for quantization. A smaller number of coordinates means less storage for model weights and less transmission overhead. The second phase is optimizing binary bases, and the third phase is optimizing the coordinates.

Post-training is the quantization with straight-through estimators (STE), promoting the overall performance compared with the result without post-training. STE propagates approximated gradient, which may introduce undesirable loss.

D. Partial Weight Averaging

SpaceDML selects partial devices randomly in a large pool and extracts their local weights—the coordinates and the binary base—to the global server. The server device multiplies the coordinates and binary bases of each device, and the sum of these multiplications becomes the weights of the global model. The details of this process are shown in Algorithm 1. Sketch initialization is implemented to generate global coordinates and binary bases before downloaded to devices. This initialization attempts to generate the same format weights in the server device as the weights in client devices. Otherwise, without this initialization, communication volume will increase, and the on-device workload, such as the local initialization for coordinates and binary bases, will increase. This random partial averaging can pick those active devices to participate in training. Any inactive and dropped devices will not suspend the process of whole federated learning.

After the server device has updated the global model, $\boldsymbol{B}$ and α on the server will be sent to the participating devices, which previously uploaded their weight to the server device. The process will be terminated when the global model reaches a target accuracy or after a predetermined number of rounds complete. If the process has not been terminated, the training will go into the next round—executing the ALQ algorithm in devices.

E. Algorithm Analysis

In common deep neural networks, we can assume $I_{0,l}$ is one for each device and layer. For n groups in layers l, after sketch initialization in each device, $I_{1,l}, I_{2,l} \ldots I_{n,l}$ stand for the bit-width of each group. We assume I_l is the average of these bit-widths. That is to say, $I_l = \frac{1}{n}\sum_{i=1}^{n} I_{i,l}$. As said in [11], after T epochs of pruning in α domain, the average bit-width I_l^T can be less than one, which is the bit-width before applying ALQ into each local model. Each device has its own global average bit-width $I^T(k)$, computing all bit-width for all layers. $I^T(k)$ is always less than 1. For VGG on CIFAR10, it can be closed to 0.4, which can largely reduce the communication amount between devices and the server. However, due to the aim of federated learning, the server can use no data. Thus, ALQ can not be used in the global center to reduce the communication overhead from server to device. All in all, communication efficiency is improved compared with the federated learning framework without any compression technique.

Algorithm 1 Quantized Weight Averaging

```
for each round t = 1, 2 ... do
    server randomly selects K devices
    server distributes B^t and α^t to K devices
    for each device k ∈ K do
        reconstruct w^t
        pretraining, ALQ quantization, post-training
        sending updated α_i and B_i to server
    end for
    server calculates: w_i^t ← B_i α_i
    server aggregates all w_i^t and obtains new global weights w^{t+1}
    server quantizes w^{t+1} as B^{t+1} and α^{t+1}
end for
```

As for the time complexity, compared with FedAvg, each device spends more time executing the ALQ algorithm. To analyze the increment time used for quantization, since on each device of SpaceDML, an independent ALQ quantization process is conducted and all ALQ quantization processes run in parallel, the increased time used for quantization of the system can be expressed by the increased time on a single device. Thus, if the increased time complexity on a single device is obtained, the total increased time complexity of SpaceDML can also be calculated. For the time complexity of ALQ on a single device, the main two steps of ALQ need to be considered. The first step is pruning in the α domain, and the second step is optimizing binary bases $\boldsymbol{B}_g$ and coordinates α_g. For the first step, the main time complexity is because of the sorting of the loss increment $f_{\alpha_{l,i}}$ caused by each $\alpha_{l,i}$. And the increased time complexity of the sorting process in the first step can be expressed as

$$O(T \cdot L \cdot card(\alpha_l) \cdot \log card(\alpha_l)),$$

where T is the total number of iterations for pruning α, and L is the number of layers of the model. Actually, $card(\alpha_i)$ is always much larger than L and T. For the second step, the time complexity for each group of each layer to optimize $\boldsymbol{B}_g$ is $O(n \cdot ((I^i_{l,m})^2 + 3I^i_{l,m} + 2))$. The time complexity used for optimizing α_g is the same as the conventional optimization step, which usually costs half of the time used for optimizing $\boldsymbol{B}_g$.

IV. Evaluation

We implement both SpaceDML and FedAvg [4] using PyTorch and compare their performances with the MNIST dataset using LeNet-5. According to the experimental results,

SPACEDML significantly reduces communication overheads of SIN using adaptive loss-aware quantization-based federated learning while keeps the same or even higher accuracies of the global models compared with FEDAVG.

Dataset and model. We use LeNet-5, a CNN that is composed of two fully connected layers and two convolution layers with max pooling, in the following experiments over the MNIST dataset. The MNIST dataset is a handwritten digits dataset ubiquitous in the AI field, consisting of 60,000 training samples and 10,000 test samples over 10 classes. Each sample in the MNIST dataset is a 28×28 grayscale image.

System settings. Both SPACEDML's system and FEDAVG's system consists of a central server and 100 devices. In our experiments, we choose 3, 5, and 10 as the number of partial devices involving in each round separately. In the implementation of SPACEDML system, we use different pruning ratios of α, which are 0.7, 0.8, and 0.9. To make the training time and computational costs of SPACEDML's devices as similar to FEDAVG's devices' as possible, in each round of SPACEDML, epoch amounts for pre-training (only in the first round), optimizing with coordinates, optimizing with basis, and post-training with STE are all one. However, to guarantee quantization performance in our system, we iteratively prune the model for four epochs. Actually, these four epochs do not lead to an unacceptable increase in training time and computational costs.

The process will be terminated as long as the round number reaches ten or the global accuracy is larger than ninety-six percent, and training samples of the MNIST dataset are uniformly distributed among all the devices, which means that each device has IID data. The batch size used by each device is 10, and in each round, both FEDAVG and SPACEDML execute the training process for one epoch.

(a) Number of chosen devices = 3.

Config	Bit-width	Acc.
FEDAVG	1	95.43
SPACEDML-0.7	0.8145	97.62
SPACEDML-0.8	0.4131	98.21
SPACEDML-0.9	0.3986	97.12

(b) Number of chosen devices = 5.

Config	Bit-width	Acc.
FEDAVG	1	95.88
SPACEDML-0.7	0.5402	98.67
SPACEDML-0.8	0.4054	98.26
SPACEDML-0.9	0.4247	96.38

(c) Number of chosen devices = 10.

Config	Bit-width	Acc.
FEDAVG	1	96.10
SPACEDML-0.7	0.5011	98.46
SPACEDML-0.8	0.5109	97.29
SPACEDML-0.9	0.4209	97.84

TABLE I: Experimental results of SPACEDML with different α's pruning ratios.

Communication overhead analysis. The experimental results of SPACEDML and FEDAVG are shown in TableI. According to the results, SPACEDML can significantly reduce the communication costs of the SIN while achieving even higher performance (2-3% in accuracy) of the global model than FEDAVG no matter the number of chosen devices in each round and the pruning ratio of α used in the system. With the increasing number of chosen devices in each round, the global model trained by FEDAVG can achieve higher accuracy. However, the increasing number of devices chosen in each round in SPACEDML sometimes doesn't impose positive impacts on the global model's accuracy, such as the results of SPACEDML-0.8 in Table (b) and Table (c). This is because of the usage of ALQ in our system. Quantization can easily introduce uncertainties to the system, which can be even more obvious when the accuracies under different scenarios are already quite high. When the number of chosen devices in each round is fixed, the increase of pruning ratio of α in SPACEDML can let the system obtain a smaller weight bit-width while also lowering the global model's accuracy. However, there are also some unexpected results in the table because of the existence of ALQ. This is because a larger pruning ratio represents that more coordinates are dropped in the optimization epoch, which makes the quantization more complete and can in turn lead to the greater loss of the precision of the reconstructed weights on the server. When the pruning ratio of α is 0.7, the lowest average weight bit-width is 0.5011 when the number of chosen devices in each round is 10, and the highest average weight bit-width is 0.8145 when the number of chosen devices in each round is 3. The difference is because the pruning epochs on each device are quite small, which means that the pruning process may be incomplete in every round when α is also small. And when the pruning ratio is 0.8 and 0.9, SPACEDML can achieve 0.39 weight bit-width, which is the best performance, and average 0.41 weight bit-width. This result is almost the optimal compression result using ALQ on a single machine, which means that SPACEDML can save at most 60% communication cost between devices and server.

Scalability analysis. Apart from saving the communication costs between devices and the server, SPACEDML can also help SINs have better scalability. This is because with the same maximum bandwidth, SPACEDML can include more devices into the system, and the results in Table also reveal thisI. Since all the devices train models with the same structure in both FEDAVG and SPACEDML, and the quantization process on each device also runs with the same setting in SPACEDML, the communication costs between each device and the server are almost the same. In this case, the overall communication overheads between devices and the server increase linearly with the increase of the devices in the system. Thus, for example, according to the result of SPACEDML-0.7 in Table (c) and the result of FEDAVG in Table (b), the overall communication costs of these two methods are almost the same since the bit-width of SPACEDML-0.7 is half of the bit-width of FEDAVG. However, the number of devices in the system of SPACEDML-0.7 is twice the number of devices in

the system of FEDAVG. Therefore, SPACEDML can improve the scalability of the SINs, helping them contain more devices.

V. CONCLUSION

This paper proposes SPACEDML, a distributed machine learning framework designed for Space Information Networks. Compared with existing federated learning frameworks, SPACEDML can significantly reduce communication bandwidth usages between client devices and the SIN global server, adaptively accommodating distributed machine learning training to SIN platforms. SPACEDML integrates the adaptive loss-aware quantization (ALQ) algorithm with distributed machine learning. This method helps the system reduce the sizes of the models' weights, which improves the efficiency and scalability of SPACEDML. We train a LeNet-5 model on the MNIST dataset with SPACEDML. Experimental results show that SPACEDML increases model accuracy by 2-3% and reduces communication bandwidth overhead by up to 60% compared with the baseline algorithm FEDAVG.

REFERENCES

[1] J. Nalepa, P. Kuligowski, M. Gumiela, M. Drobik, and M. Nowak, "Leopard: A New Chapter in On-board Deep Learning-powered Analysis of Hyperspectral Imagery," in *IAC - The CyberSpace Edition*, 2018.

[2] A. Cesta, G. Cortellessa, M. Denis, A. Donati, S. Fratini, A. Oddi, N. Policella, E. Rabenau, and J. Schulster, "Mexar2: AI Solves Mission Planner Problems," *IEEE Intelligent Systems*, vol. 22, no. 4, pp. 12–19, 2007.

[3] J. Manning, D. Langerman, B. Ramesh, E. Gretok, C. Wilson, A. George, J. MacKinnon, and G. Crum, "Machine-Learning Space Applications on SmallSat Platforms with TensorFlow," in *AIAA/USU CSS*, 2018.

[4] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient Learning of Deep Networks from Decentralized Data," in *AIS*, 2017.

[5] J. Du, C. Jiang, Q. Guo, M. Guizani, and Y. Ren, "Cooperative Earth Observation Through Complex Space Information Networks," *IEEE Wireless Communication*, vol. 23, no. 2, pp. 136–144, 2016.

[6] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtarik, A. T. Suresh, and D. Bacon, "Federated Learning: Strategies for Improving Communication Efficiency," in *NeurIPS Workshop on Private Multi-Party Machine Learning*, 2016.

[7] A. Li, J. Sun, B. Wang, L. Duan, S. Li, Y. Chen, and H. Li, "LotteryFL: Personalized and Communication-efficient Federated Learning with Lottery Ticket Hypothesis on Non-IID Datasets," *arXiv preprint arXiv:2008.03371*, 2020.

[8] D. Rothchild, A. Panda, E. Ullah, N. Ivkin, I. Stoica, V. Braverman, J. Gonzalez, and R. Arora, "FetchSGD: Communication-efficient Federated Learning with Sketching," in *ICML*, 2020.

[9] M. Courbariaux, Y. Bengio, and J.-P. David, "BinaryConnect: Training Deep Neural Networks with Binary Weights during Propagations," *arXiv preprint arXiv:1511.00363*, 2015.

[10] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1," *arXiv preprint arXiv:1602.02830*, 2016.

[11] Z. Qu, Z. Zhou, Y. Cheng, and L. Thiele, "Adaptive Loss-aware Quantization for Multi-bit Networks," in *CVPR*, 2020.

[12] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via Gradient Quantization and Encoding," *arXiv preprint arXiv:1610.02132*, 2016.

[13] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "FedPAQ: A Communication-efficient Federated Learning Method with Periodic Averaging and Quantization," in *ICAIS*, 2020.

[14] N. Shlezinger, M. Chen, Y. C. Eldar, H. V. Poor, and S. Cui, "Federated Learning with Quantization Constraints," in *ICASSP*, 2020.

[15] M. M. Amiri, D. Gunduz, S. R. Kulkarni, and H. V. Poor, "Federated Learning with Quantized Global Model Updates," *arXiv preprint arXiv:2006.10672*, 2020.