Sparsification and Optimization for Energy-Efficient Federated Learning in Wireless Edge Networks

Lei Lei[1], Yaxiong Yuan[2], Yang Yang[3], Yu Luo[4], Lina Pu[5], and Symeon Chatzinotas[2]

[1]School of Information and Communications Engineering, Xi'an Jiaotong University, China
[2]Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, Luxembourg
[3] Competence Center for High Performance Computing Fraunhofer ITWM, Germany
[4]Department of Electrical and Computer Engineering, Mississippi State University, USA
[5]Department of Computer Science, University of Alabama, USA

***Abstract*—Federated Learning (FL), as an effective decentralized approach, has attracted considerable attention in privacy-preserving applications for wireless edge networks. In practice, edge devices are typically limited by energy, memory, and computation capabilities. In addition, the communications between the central server and edge devices are with constrained resources, e.g., power or bandwidth. In this paper, we propose a joint sparsification and optimization scheme to reduce the energy consumption in local training and data transmission. On the one hand, we introduce sparsification, leading to a large number of zero weights in sparse neural networks, to alleviate devices' computational burden and mitigate the data volume to be uploaded. To handle the non-smoothness incurred by sparsification, we develop an enhanced stochastic gradient descent algorithm to improve the learning performance. On the other hand, we optimize power, bandwidth, and learning parameters to avoid communication congestion and enable an energy-efficient transmission between the central server and edge devices. By collaboratively deploying the above two components, the numerical results show that the overall energy consumption in FL can be significantly reduced, compared to benchmark FL with fully-connected neural networks.**

I. Introduction

Machine learning is becoming an important tool for next-generation wireless communication networks. In practical applications, most machine learning schemes follow a centralized training framework, where the collected local data is transmitted to a central server for processing and training. However, most of the collected data is privacy-sensitive and such data exchange may have potential risks in data leakage. To this end, federated learning (FL), as a distributed learning paradigm, has attracted considerable attention in many wireless applications, such as wireless federated edge learning systems.

FL enables edge devices to collaboratively learn a global learning model while keeping all the training data on the devices. The training process of FL is iterative. At each learning cycle, each local user trains the local models based on the collected data and the current global model. Then, all the edge devices upload the local model to a centralized server without the requirement for the raw data. After that, the centralized server will broadcast an aggregated global model to all the edge devices for the next learning cycle.

978-1-6654-3540-6/22 © 2022 IEEE

A. Related Works

In the literature, federated stochastic gradient descent (FedSGD) and federated averaging (FedAvg) are the most well-known FL algorithms, which minimize the local loss functions in each device and update the learning models by stochastic gradient descent (SGD) [1]. Compared to FedSGD, FedAvg increases the number of local training epochs performed on the edge device to reduce the frequency of communication between the server and edge devices [2].

Recent works have applied FL to wireless communication networks. In [3], the authors developed an edge learning framework to analyze the convergence performance of the FL algorithm while considering the features of wireless networks, e.g., user scheduling policy, channel fading and inter-cell interference. In [4], a joint user selection and wireless resource allocation problem was investigated to minimize the loss function with limited transmit power. To minimize FL training time in multiple-input multiple-output systems, in Considering the scarcity of energy in the local devices of wireless federated edge learning networks, the authors in [5] and [6] formulated energy minimization problems, including local computing energy and transmission energy, via optimizing wireless and computation resources. Furthermore, the authors in [6] proposed an improved FL algorithm based on FedProx to handle the non-i.i.d. data and heterogeneity of the devices. In [7], an optimization problem was formulated to minimize to capture the trade-off between FL convergence time and energy consumption of UEs with heterogeneous computing and power resources.

In most of the previous works, e.g., [3]–[7], the adopted learning models are typically fully-connected deep neural networks (DNNs). Such heavyweight learning models might not be the best choice for local training in edge devices. This is because a massive number of weight parameters need to be uploaded/downloaded repeatedly, and thus could result in heavy overhead, slow convergence, and high energy consumption. Besides, the widely-adopted FL algorithms [1], [2] may not guarantee the performance in non-smooth cases, e.g., adding L1-norm regularization for DNN sparsification.

B. Contributions

Beyond the state-of-the-art, we are motivated to develop a lightweight learning model to relieve the computation burden and develop a suited algorithm for local training. In this work, we propose a joint sparsification and optimization scheme for wireless federated edge learning. The main contributions are summarized as follows:

- Instead of adopting widely-used DNNs, we introduce sparse neural networks (SNN) to simplify the computational operations in local training and reduce the volume of transmitted data in communications.
- We formulate an energy minimization problem via optimizing wireless resources and learning parameters. Different from the energy models adopted in previous works, we jointly minimize computing energy, uploading energy, and broadcasting energy. All three types of energy consumption are non-trivial for wireless edge networks.
- We design a joint energy-saving scheme that integrates model sparsification in local training and resource optimization in communications, such that the energy conservation can benefit from multiple aspects.
- Numerical results verify that the joint sparsification and optimization scheme reduces more than 30% energy consumption compared to benchmark FL schemes, where resource optimization brings 16.43% performance gain on energy saving, and sparsification contributes to 15% performance improvement.

The remainder of this paper is organized as follows. In Section II, we provide the preliminary of FL framework and energy models. Section III introduces the learning model sparsification, the formulated energy minimization problem, and the proposed joint scheme. Numerical results are presented and analyzed in Section IV. Finally, we draw the conclusions in Section V.

II. FL Framework and Energy Models

A. FL Framework

FL is designed to allow multiple devices to cooperatively execute a learning task by only uploading the local learning model to a central server that can be co-located with base stations or access points. The edge devices with certain computing capabilities train local models based on the collected data. The objective of FL is given by:

$$\min_{\mathbf{w}\in\mathcal{W}} F(\mathbf{w}) = \sum_{k=1}^{K} \frac{D_k}{D} F_k(\mathbf{w}), \tag{1}$$

where $\mathbf{w}$ is the parameter vector of the learning model, $\mathcal{W}$ is the space of the parameters, K is the number of the edge devices participating in the training, D_k is the size of the training data at edge device k, and D is the total size of the training data. The local objective function $F_k(\mathbf{w})$ consists of a loss function $f_k(\mathbf{w})$ and a regularization item $r_k(\mathbf{w})$, which can be expressed as:

$$F_k(\mathbf{w}) = \underbrace{\frac{1}{D_k}\sum_{l=1}^{D_k} \mathcal{L}_{k,l}(\mathbf{w})}_{\triangleq f_k(\mathbf{w})} + r_k(\mathbf{w}), \tag{2}$$

where $\mathcal{L}_{k,l}(\mathbf{w})$ describes the prediction error of the l-th single sample at edge device k, which can be represented by mean square error (MSE) or cross entropy. The regularization item $r_k(\mathbf{w})$ is a penalty to prevent overfitting or confining the weight values.

The process of the FL is conducted in an iterative manner as the following steps [1], [2]:

- Step 1: The centralized server randomly selects edge devices to participate in the training process and broadcasts a global FL model.
- Step 2: Each selected edge device downloads the global FL model for training a local FL model. The well-trained local models will be uploaded to the centralized server.
- Step 3: The centralized server aggregates the local information to update the global learning model.
- Step 4: Repeating steps 1-3 (i.e., a learning cycle) until the termination condition is met.

In an FL framework, problem (1) is decomposed into K independent problems that are solved locally at each edge device. Thus, each device trains the local learning model by minimizing the local loss function:

$$\min_{\mathbf{w}\in\mathcal{W}} : F_k(\mathbf{w}). \tag{3}$$

The local problems can be solved by stochastic gradient descent (SGD) algorithm, which updates the local parameters iteratively and consumes local iterations. We denote i as the index of global iterations (or learning cycles), and j as the index of local iterations. At each local iteration, taken the received global model $\mathbf{w}^i$ as the initial point, the update rule is written by:

$$\mathbf{w}_k^{i,j+1} = \mathbf{w}_k^{i,j} - \varepsilon_{loc} \nabla F_k(\mathbf{w}_k^{i,j}). \tag{4}$$

where ε_{loc} is the local update step and $\mathbf{w}_k^{i,j}$ is the local FL model for user k at the j-th local iteration and the i-th global iteration. We assume that the SGD terminates at the maximum number of local iterations, denoted by J_k. The server updates the global model by:

$$\mathbf{w}^{i+1} = \mathbf{w}^i + \epsilon \left(\sum_{k=1}^{K} \mathbf{w}_k^i - \mathbf{w}^i \right), \tag{5}$$

where ϵ is the learning rate and $\mathbf{w}_k^i = \mathbf{w}_k^{i,J_k}$. For SGD algorithm, the gap between the optimum and the intermediate result at local iteration j is expressed as:

$$\Gamma(j,d) = \alpha \left(\frac{1}{\sqrt{jd}} + \frac{1}{j} \right), \tag{6}$$

where d is the size of training data and α is a positive

constant. We remark that a small gap $\Gamma(j,d)$ means high local learning accuracy. Since the local model needs to be uploaded to the server for global aggregation, the local accuracy directly affects the global accuracy of FL. To avoid the performance degradation caused by the accumulation error of local computing, the gap $\Gamma(j,d)$ should be less than a threshold γ_{th}.

B. Energy Models

1) Communication Energy: At each global iteration (or learning cycle), the channel gain for edge device k is h_k, $\forall k \in \{1,...,K\}$. Under the assumption of the block fading channel, the channel states keep constant within a learning cycle. Considering the wireless edge scenarios, the server for edge learning could be with limited energy. The energy consumed on both edge devices' upload and server's broadcast needs to be conserved. To facilitate the data aggregation in the server, the learning models for all the edge devices are identical. We denote A_k as (in bits) the upload data volume of edge device k and A_0 (in bits) as the broadcast data volume of the server.

In the upload phase, the transmission time of edge device k within a learning cycle is:

$$T_k^u = \frac{A_k}{R_k} = \frac{A_k}{B_k \log(1+\frac{p_k h_k}{\sigma^2})}, \tag{7}$$

where R_k, B_k, p_k are the uplink transmission rate, the uplink bandwidth, and the transmit power of edge device k, respectively, and σ^2 is the noise power. The total energy consumption E^u for all the edge devices can be expressed as:

$$E^u = \sum_{k=1}^{K} p_k T_k^u. \tag{8}$$

In the broadcast phase, since the devices' transmission rates are different, the server broadcasts the global FL model until the last edge device receives the data. Thus, the broadcast time T_0^d can be calculated by:

$$T_0^d = \max_k \left\{T_k^d\right\} = \max_k \left\{ \frac{A_0}{B_0 \log(1+\frac{p_0 h_k}{\sigma^2})} \right\}, \tag{9}$$

where T_k^d is the transmission time consumed from server to edge device k, and B_0 is the bandwidth of server. We denote p_0 as the transmit power of the server, the broadcast energy E^d is given by:

$$E^d = p_0 T_0^d. \tag{10}$$

2) Computation Energy: The CPU power dissipation includes static and dynamic power, where static power that arises from bias and leakage currents can be dramatically reduced by careful hardware design and dominant CPU power consumption is therefore dynamic power. According to [8], the dynamic power of a CPU P_k^c for edge device k depends on the CPU's supply voltage V_k and CPU computation capacity f_k (the number of CPU cycles/second), i.e., $P_k^c \propto V_k^2 f_k$. In addition, another relation in CPU operation is that f_k is positively proportional to V_k, i.e., $V_k \propto f_k$. Thus, the computation power for edge device k can be written as $P_k^c = \kappa f_k^3$, where κ is the effective switched capacitance depending on the chip architecture. We denote, for edge device k, J_k is the maximum number of local iterations, C_k is the number of CPU cycles required for processing a single sample data, and D_k is the number of collected training samples. The processing time needed can be expressed as:

$$T_k^c = \frac{C_k J_k D_k}{f_k}, \tag{11}$$

and the total computation energy for local data processing within a learning cycle is given by:

$$E^c = \sum_{k}^{K} P_k^c T_k^c = \sum_{k}^{K} \kappa J_k C_k D_k f_k^2 = \sum_{k=1}^{K} M_k D_k. \tag{12}$$

where $M_k = \kappa I_k C_k f_k^2$. The computation energy at the server can be ignored as the aggregation operation is simple, i.e., averaging [5].

In the timeline of the FL process, firstly, the edge devices that have received the broadcast data can perform local calculations immediately. We consider a half-duplex mode. The server cannot receive and transmit data simultaneously. Before uploading the local FL model to the server, the edge devices need to confirm the server has completed the broadcast task, e.g., receive a control signal allowing to upload. The edge devices that finish the local calculations faster may need to wait for the other devices. Therefore, the time when edge device k starts uploading local learning model is $\max\{T_k^d + T_k^c, T_0^d\}$, where T_k^d, T_0^d and T_k^c can be obtained from (9) and (11). When all the edge devices complete their upload tasks, we can calculate the total time of a learning cycle by:

$$T_{tot} = \max_k \{T_k^u + \max\{T_k^d + T_k^c, T_0^d\}\} \tag{13}$$

III. Sparsification and Optimization For Energy-Efficient FL

We propose a joint scheme to minimize consumed energy in FL via: 1) applying a sparse learning model to reduce the transmission data and the computation operations; and 2) determining the energy-efficient wireless resource allocation and learning configurations by optimization.

A. Sparsification for Learning Models

From (9) and (10), the communication energy can be reduced when A_k decreases. In addition, based on (12), the computation energy increases with C_k. This motivates us to implement a learning model with fewer weight vectors and computation operations. In practice, the learning models with fewer weights or simpler constructions are sufficient to extract the data features with acceptable learning accuracy [9]. Sparse NN (SNN) is one of the methods for simplification. It removes the majority of weights (force them to 0) such that only a percentage of the possible connections exist between layers.

To realize SNN, we set regularization items after the loss function such as L1-norm in (14) or L2-norm in (15).

$$\min_{\mathbf{w}\in\mathbb{R}^n} F_k(\mathbf{w}) = f_k(\mathbf{w}) + \mu\|\mathbf{w}\|, \tag{14}$$

$$\min_{\mathbf{w}\in\mathbb{R}^n} F_k(\mathbf{w}) = f_k(\mathbf{w}) + \frac{\mu}{2}\|\mathbf{w}\|^2, \tag{15}$$

where μ is the regularization rate selected from [0,1]. To obtain a model with optimal performance, the regularization rate μ should be properly tuned and cannot be excessively large.

In order to quantitatively analyze the energy-saving gain brought by the sparse models, we calculate A_k and C_k for DNN and SNN. For edge device k, we define Θ_k as the average unit bits corresponding to the numeric data type of the weights and Ψ_k as the total number of the weights of the learning model. Thus, A_k, as the total data volume, is calculated by:

$$A_k = \Theta_k \cdot \Psi_k. \tag{16}$$

We consider DNN and SNN have the same structure with L layers and x_l neurons at the l-th layer, so that Ψ_k is the same for the three models, expressed as $\sum_{l=1}^{L} x_{l-1}x_l + x_l$. The numeric type of the parameter in DNN is floating-point with $\Theta_k = 32$ (bits). For SNN, we assume $\rho_k^1(\%)$ of the weights are zero and the others are floating-point numbers. Therefore, the average unit bits for SNN are $\Theta_k = 32(1-\rho_k^1)$.

To calculate the CPU cycles per sample C_k, we first introduce Φ_k as the required number of floating-point operations (FLOPs) to process a single training sample, where an addition, multiplication, or division is defined as a FLOP. For DNN and SNN, they both need $2\Psi_k$ addition and multiplication operations [10]. The number of CPU cycles needed for each operation is fixed, denoted by e_k, which depends on the performance of the edge device. The number of CPU cycles required for a single training sample C_k is expressed as:

$$C_k = \Phi_k \cdot e_k. \tag{17}$$

B. Resource Optimization in FL

Based on the energy models and the sparse learning models, we further reduce the total energy by formulating an optimization problem. The edge devices, such as smartphones or small-sized sensors, have low transmit power and limited ability to adjust the uplink power. Thus, to minimize the uplink energy, we optimize bandwidth B_k and with a fixed transmit power p_k. For the downlink transmission, all the edge devices receive the broadcasted data via the shared spectrum. To improve the data rate, we set B_0 as the maximum available bandwidth of the server. The downlink energy can be optimized by adjusting transmit power p_0. In addition, we minimize the computation energy by selecting a proper training data size D_k. The minimization problem for a learning cycle is formulated as:

$$\mathbf{P0}: \min_{p_0, B_k, D_k} E^u + E^d + E^c \tag{18a}$$

$$s.t.\ T_{tot} \le T_{th}, \tag{18b}$$

$$M_k D_k \le E_{max}^C, \forall k, \tag{18c}$$

$$\Gamma(J_k, D_k) \le \gamma_{th}, \quad \forall k, \tag{18d}$$

$$\sum_{k=1}^{K} R_k \le R_{cap}, \tag{18e}$$

$$\sum_{k=1}^{K} b_k \le B_{tot}, \tag{18f}$$

$$p_0 < P_{max}. \tag{18g}$$

The objective function (18a) in $\mathbf{P0}$ includes three components, i.e., local computing energy, uploading energy, and broadcasting energy. In constraint (18b), the time consumption at each learning cycle T_{tot} should be less than T_{th}. The computation energy for each device should be less than a limit value E_{max}^C, as expressed in (18c), due to the limited battery storage. Towards minimizing the local computing energy, the optimizer tends to reduce D_k from (12). However, a smaller D_k leads to a large gap $\Gamma(J_k, D_k)$ and the learning accuracy deteriorates. To guarantee the learning accuracy of FL, the constraint (18d) confines that the local computing gap should be under the threshold γ_{th}. The constraints (18e) represent the total uplink rate should not excess the uplink capacity R_{cap}. The constraints (18f) and (18g) are the limitations of the spectrum and power, respectively.

As the original problem $\mathbf{P0}$ contains the maximum operators, we reformulate $\mathbf{P0}$ to an equivalent problem $\mathbf{P1}$ with auxiliary variables z_k and T_0^d.

$$\mathbf{P1}: \min_{\substack{p_0, B_k, D_k, \\ z_k, T_0^d}} \sum_{k=1}^{K} \frac{W p_k}{B_k \log(1+\frac{p_k h_k}{\sigma^2})} + p_0 T_0^d + \sum_{k=1}^{K} M_k D_k$$

$$s.t. \frac{W}{B_0 \log(1+\frac{p_0 h_k}{\sigma^2})} \le T_0^d, \quad \forall k, \tag{19a}$$

$$\frac{W}{B_0 \log(1+\frac{p_0 h_k}{\sigma^2})} + \frac{M_k}{f_k} \le z_k, \quad \forall k, \tag{19b}$$

$$z_k + \frac{W_k}{B_k \log(1+\frac{p_k h_k}{\sigma^2})} \le T_{th}, \quad \forall k, \tag{19c}$$

$$M_k D_k \le E_{max}^C, \forall k, \tag{19d}$$

$$\alpha\left(\frac{1}{\sqrt{J_k D_k}} + \frac{1}{J_k}\right) \le \gamma_{th}, \quad \forall k, \tag{19e}$$

$$T_0^D \le z_k, \quad \forall k, \tag{19f}$$

$$(18e), (18f), (18g).$$

$\mathbf{P1}$ is a non-convex problem because of the bilinear item $p_0 T_0^d$. To solve the problem, the bilinear term can be relaxed and bounded by McCormick envelopes [11]. We remark that $\mathbf{P1}$ is solved offline at the beginning of each learning cycle for the optimal network and learning configuration.

C. The Proposed Joint Scheme

The joint energy-saving scheme is summarized in Alg. 1 including global and local iterations. In a global iteration (lines 5-19, also defined as a learning cycle), the server randomly selects K edge devices in line 6. In line 7, the optimization problem $\mathbf{P1}$ is solved to determine the wireless resource allocation, i.e., the server's transmit power, uplink bandwidth, and the size of training data. With the optimized solutions of

$\mathbf{P1}$, the server allocates resources and broadcasts the global learning model in line 8. Each edge device then trains a local model. After receiving the local models, the server can update the global model in lines 17-18.

Algorithm 1 Joint sparsification and optimziation scheme

Input:

1: Initial global FL model: $\mathbf{w}^0$;
2: Maximum number of global iteration: I;
3: Number of local iterations for each edge device: J_k;
4: Learning rate: ϵ.
5: **for** $i = 0 : I$ **do**
6: Server selects K devices randomly.
7: Server determines the optimal resource allocation and learning parameter by solving $\mathbf{P1}$.
8: Server sends $\mathbf{w}^i$ to all chosen devices.
9: **for** $k = 1 : K$ (do in parallel) **do**
10: **for** $j = 0 : J_k$ **do**
11: Edge device selects minibatch of collected data.
12: Edge device calculates momentum by (21).
13: Edge device solves sub-problem (22).
14: Edge device updates $\mathbf{w}_k^{i,j}$ by (23).
15: **end for**
16: **end for**
17: All the devices send $\mathbf{w}_k^i$ back to the server.
18: Server updates $\mathbf{w}^i$ by (5).
19: **end for**

Output: Well-trained local model $\mathbf{w}_k^*$.

The local iteration (lines 10-15) refers to the local training process performed at edge devices. We aim at solving the local loss function (3). To address the non-smoothness in SNN-based FL, e.g., L1-norm regularization $\mu\|\mathbf{w}\|$, we extend the previous proposed ProxSGD in [12] to the FL framework. The former is designed for centralized learning with non-smooth loss functions. Firstly, in line 11, a minibatch $\mathcal{M}^{i,j}$ is randomly selected from the training data to estimate the gradient of $f_k(\mathbf{w})$, i.e.,

$$\bar{\nabla} f_k(\mathbf{w}^{i,j}) = \frac{1}{|\mathcal{M}^{i,j}|} \sum_{l \in \mathcal{M}^{i,j}} \nabla \mathcal{L}_{k,l}(\mathbf{w}^{i,j}). \tag{20}$$

To keep the gradient updated in the right direction and accelerate the converge speed, in line 12, we introduce momentum $\mathbf{v}_k^{i,j}$, which is formed in a recursive manner:

$$\mathbf{v}_k^{i,j} = (1 - \varrho^j)\mathbf{v}_k^{i,j-1} + \varrho^j \bar{\nabla} f_k(\mathbf{w}^{i,j}), \tag{21}$$

where $\varrho^j \in (0, 1]$ is the step size for the momentum. In line 13, we solve an approximation sub-problem and obtain an intermediate solution $\hat{\mathbf{w}}^{i,j}$:

$$\hat{\mathbf{w}}_k^{i,j} = \underset{\mathbf{w} \in \mathbb{R}^n}{\operatorname{argmin}} \ (\mathbf{w} - \mathbf{w}^{i,j})^{\mathrm{T}} \mathbf{v}_k^{i,j} + \frac{\tau}{2}\|\mathbf{w} - \mathbf{w}^{i,j}\|^2 + r_k(\mathbf{w}). \tag{22}$$

The objective in (22) is a convex approximation of $F_k(\mathbf{w})$ around the point $\mathbf{w}^{i,j}$ by incorporating first-order Taylor expansion, quadratic regularization and non-smooth term. Compared to the conventional SGD, the update rule in (22) can resolve the issues of weights' constraints and non-smooth (non-differentiable) objectives. The difference between $\hat{\mathbf{w}}^{i,j}$ and $\mathbf{w}^{i,j}$ specifies the update direction which is used to refine the parameters:

$$\mathbf{w}_k^{i,j+1} = \mathbf{w}_k^{i,j} + \varepsilon_{loc}(\hat{\mathbf{w}}_k^{i,j} - \mathbf{w}_k^{i,j}). \tag{23}$$

When the local training terminates at the J_k-th iteration, each edge device uploads the local learning model to the server. The server collects all the information and updates the global model by (5). The algorithm terminates when reaching the maximal number of global iterations I.

IV. Simulation Results

In this section, we evaluate the performance of the proposed sparsification-and-optimization scheme. We compare the performance of DNN and SNN, in terms of the distribution of weight values and the total energy consumption. The adopted parameters in simulations are summarized in Table I.

Table I: Parameter settings

Data set	MNIST
Total number of devices	1000
Number of selected devices per learning cycle K	10
Transmit power on edge device p_k	0.5 Watt
Bandwidth of server B_0	3 MHz
SINR	15-23 dB
Input size	28× 28 images
Number training data samples	69035
Loss function	cross entropy
learning rate ϵ	0.001
Number of global iterations I	500
Batch size	50
Number of local iterations J_k	50
Penalty parameter μ	0.0025 and 0.001
Computation capacity f_k	6.725 GHz
FLOPs per CPU cycle e_k	4
Effective switched capacitance κ	10^{-28}

A. Performance on Energy Conservation

To demonstrate the impact of different learning models on energy consumption, we evaluate the cumulative distribution function (CDF) of weight values and show the total energy consumption of DNN and SNN in Fig. 1 and 2, respectively. From Fig. 1, in DNN, around 80% weights are evenly distributed in the interval [-0.2, 0.2], and few weights are with $\|\mathbf{w}\| < 0.01$. Thus, all the weights in DNN should be uploaded or broadcasted as floating-point values. For SNN, when μ=0.001, around 55% of the weights are exactly equal to 0, such that only 45% of the weight needs to participate in the communication. When μ drops to 0.00025, the impact of regularization reduces with 35% zero-valued weights.

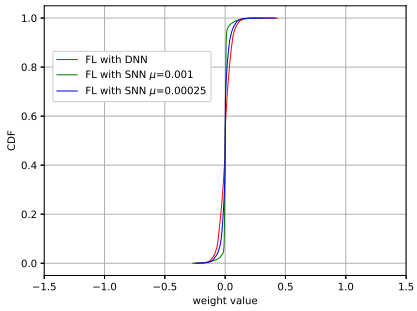

Fig. 1. Distribution of weights of different neural networks.

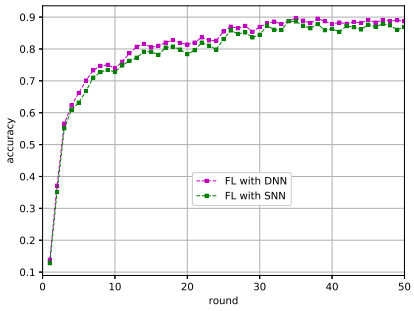

Fig. 3. Accuracy of different neural networks.

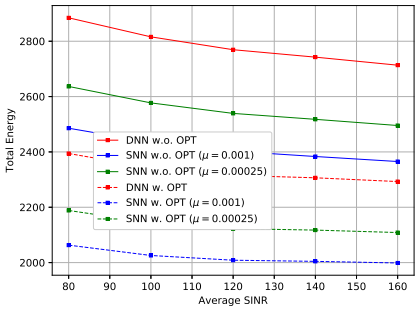

Fig. 2. Total energy vs. SINR.

In Fig. 2, we show the total energy consumption among different FL schemes with regard to the average SINR. Generally, the total energy consumption decreases with a higher value of SINR. This is because, in a better channel condition, the transmission rate is higher and the consumed energy can be saved with less transmission time. The proposed scheme with SNN (μ=0.001) and resource optimization saves around 30% energy consumption compared to DNN-based FL scheme without energy optimization. Specifically, SNN (μ=0.001) saves 15.72% total energy compared to DNN, the optimization part further reduces energy consumption by 16.43%. In addition, SNN with μ=0.001 consumes 6.2% less energy than SNN with μ=0.00025, but a higher μ may result in lower learning accuracy.

B. Performance on Learning Accuracy

Fig. 3 shows the learning accuracy of adopting different learning models. The learning task is with the identical learning rate and training data. The accuracy performance in DNN and SNN (μ=0.001) maintains at the same level, though DNN slightly outperforms, 89% against 87%. Recalling Fig. 2, SNN simplifies the local model and reduces the energy. This implies that the adopted sparsification scheme achieves better trade-off performance than DNN-based FL.

V. Conclusion

In this paper, we investigated an energy-efficient federated scheme implemented in wireless federated edge learning networks to economize energy from two perspectives. Firstly, we formulated an optimization problem to minimize the energy consumption, including communication and computation energy, via wireless resource management and learning parameter allocation. Secondly, based on the analysis of the energy consumption of different learning models, the energy can be further saved by selecting sparse instead of traditional DNN. Numerical results show considerable energy-saving gains of the sparsification and optimization scheme.

References

[1] K. Thonglek, K. Takahashi, K. Ichikawa, H. Iida, and C. Nakasan, "Federated Learning of Neural Network Models with Heterogeneous Structures," in Proc. *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 735-740, Dec. 2020.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in Proc. *International Conference on Artificial Intelligence and Statistics (AISTATS)*, Apr. 2017.

[3] H. H. Yang, Z. Liu, T. Q. Quek, and H. V. Poor, "Scheduling policies for federated learning in wireless networks," *IEEE Transactions on Communications*, vol. 68, no. 1, pp. 317-333, Sept. 2019.

[4] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A Joint Learning and Communications Framework for Federated Learning Over Wireless Networks," in *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 269-283, Jan. 2021.

[5] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy Efficient Federated Learning Over Wireless Communication Networks," in *IEEE Transactions on Wireless Communications*, vol. 20, no. 3, pp. 1935-1949, Mar. 2021.

[6] V. D. Nguyen, S. K. Sharma, T. X. Vu, S. Chatzinotas, and B. Ottersten, "Efficient federated learning algorithm for resource allocation in wireless iot networks," in *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3394-3409, Sept. 2020.

[7] C. T. Dinh, N. H. Tran, M. N. Nguyen, C. S. Hong, W. Bao, A. Y. Zomaya, and V. Gramoli, "Federated learning over wireless networks: Convergence analysis and resource allocation," in *IEEE/ACM Transactions on Networking,* vol. 29, no. 1, pp. 398-409, Nov. 2020.

[8] I. Ahmad and S. Ranka eds, "Handbook of Energy-Aware and Green Computing, Volume 1", CRC Press, Jan. 2012.

[9] I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning," MIT press, 2016.

[10] Y. Yuan, L. Lei, T. X. Vu, S. Chatzinotas, S. Sun, and B. Ottersten, "Energy minimization in UAV-aided networks: actor-critic learning for constrained scheduling optimization," in *IEEE Transactions on Vehicular Technology,* vol. 70, no. 5, pp. 5028-5042, May 2021.

[11] G. P. McCormick, "Computability of Global Solutions to Factorable Nonconvex Programs: Part I Convex Underestimating Problems," in *Mathematical Programming,* vol. 10, no. 1, pp. 147-175, Dec. 1976.

[12] Y. Yang, Y. Yuan, A. Chatzimichailidis, R. J. van Sloun, L. Lei, S. Chatzinotas, "Proxsgd: Training structured neural networks under regularization and constraints,' in Proc. *International Conference on Learning Representations (ICLR)*, Sept. 2019.