2024 IEEE International Performance, Computing, and Communications Conference (IPCCC) | 979-8-3503-6794-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/IPCCC59868.2024.10850123

# The Robustness of Spiking Neural Networks in Communication and its Application towards Network Efficiency in Federated Learning

Manh V. Nguyen*, Liang Zhao*, Bobin Deng*, William Severa†, Honghui Xu*, Shaoen Wu*
*College of Computing and Software Engineering, Kennesaw State University, Georgia, USA
mnguy126@students.kennesaw.edu, {lzhao10, bdeng2, hxu10, swu10}@kennesaw.edu
†Department of Cognitive & Emerging Computing, Sandia National Laboratories, New Mexico, USA
wmsever@sandia.gov

***Abstract*—Spiking Neural Networks (SNNs) have recently gained significant interest in on-chip learning in embedded devices and emerged as an energy-efficient alternative to conventional Artificial Neural Networks (ANNs). However, to extend SNNs to a Federated Learning (FL) setting involving collaborative model training, the communication between the local devices and the remote server remains the bottleneck, which is often restricted and costly. In this paper, we first explore the inherent robustness of SNNs under noisy communication in FL. Building upon this foundation, we propose a novel Federated Learning with Top-$\kappa$ Sparsification (FLTS) algorithm to reduce the bandwidth usage for FL training. We discover that the proposed scheme with SNNs allows more bandwidth savings compared to ANNs without impacting the model's accuracy. Additionally, the number of parameters to be communicated can be reduced to as low as $6\%$ of the size of the original model. We further improve the communication efficiency by enabling dynamic parameter compression during model training. Extensive experiment results demonstrate that our proposed algorithms significantly outperform the baselines in terms of communication cost and model accuracy and are promising for practical network-efficient FL with SNNs.**

***Index Terms*—Federated Learning, Spiking Neural Networks**

## I. Introduction

Spiking Neural Networks (SNNs) constitute a significant advancement in artificial intelligence (AI), operating with asynchronous discrete events called spikes, which enhances energy efficiency, particularly for neuromorphic hardware designed to mimic the brain's neural architecture. This efficiency is crucial as AI's energy demands rise, positioning SNNs as a solution to the impending energy crisis [1]. Beyond neuromorphic hardware, SNNs are also suitable for general-purpose use, including on resource-constrained edge devices with limited power budgets. Studies and practical applications have demonstrated that they provide significant energy and resource efficiency [2], [3], making SNNs an attractive option for sustainable, low-power AI systems across various domains, from industrial automation to smart home technologies.

For efficiency and privacy concerns, Federated Learning (FL) has emerged as an effective framework for distributed AI systems. It has great potential to be integrated with SNNs [4] to enjoy the benefits of both worlds. With FL approaches, instead of aggregating the raw data from distributed entities to a central server, the model is trained across multiple devices, each holding local data samples. The central server then aggregates the updates of the locally trained models. However, FL training methods also incur significant communication costs due to frequent updates and exchanges of model parameters between the central server and the decentralized nodes. This makes communication the bottleneck in the large-scale deployment of FL systems. Additionally, most existing FL frameworks assume perfect communication while real-world communication networks are noisy and prone to packet loss and transmission errors [5]. Noisy communication can lead to corrupted data exchanges, negatively impacting the model's convergence and accuracy. Addressing these issues is crucial for the widespread adoption and efficiency of FL systems.

This work attempts to improve network efficiency in FL with SNNs considering noisy communication channels. We first explore the robustness of SNNs compared to Artificial Neural Networks (ANNs) as a foundation for its adaptation in bandwidth-limited environments. By leveraging the robustness of SNNs to noise, we propose a suite of Top-$\kappa$ Sparsification based algorithms to reduce bandwidth consumption in FL with SNN. Finally, we conduct empirical experiments to verify the superiority of SNNs to ANNs in FL settings under noisy communication scenarios. Furthermore, we evaluate the effectiveness of the proposed algorithms in terms of communication cost and accuracy and their sensitivity to network size and communication compression rate. This work aims to provide practical and advantageous FL solutions for edge-device SNNs, whose environments are often under limited and noisy communication settings.

**Contribution.** The key contribution of the work presented in this paper is three-fold:

- To the best of our knowledge, we are among the first to investigate the performance of FL with SNN under noisy communication. We discover that SNN is significantly more robust than equivalent ANN models.
- In light of the inherent robustness of SNN, we propose Federated Learning with Top-$\kappa$ Sparsification to improve communication efficiency. Additionally, we leverage the

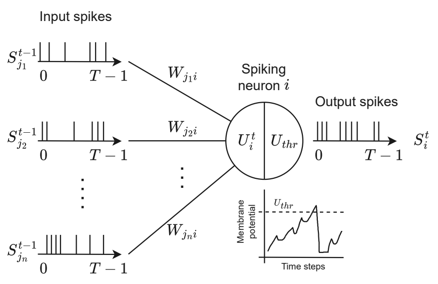


Fig. 1. Processing Mechanism of Integrate-and-Fire (IF) Spiking Neuron.

principle of critical learning periods in Federated Learning and propose a Federated Learning with Dynamic-$\kappa$ Reduction to further reduce the communication overhead.

- We conduct extensive experiments to validate the proposed algorithms. The results demonstrate that the novel algorithms enable SNN to save significant bandwidth compared to ANN in FL settings with noisy communication. Further, they can achieve comparable model accuracy with 6% of the communication cost compared to the baselines.

## II. Background of Federated Learning and Spiking Neural Networks

In this section, we briefly cover the background of Spiking Neural Networks (SNNs) and Federated Learning (FL).

### A. Spiking Neural Network

**Rate encoding**. A spiking neuron is modeled to perceive input as incoming spikes over a predefined time interval. The neuron accumulates membrane potential while absorbing the input spikes and scales by the synaptic weights. Once the membrane potential reaches a certain threshold, the neuron fires an output spike, releases the membrane energy, and the process restarts. This mechanism is known as Integrate-and-Fire (IF) and is illustrated in Fig 1. Another popular type of spiking neuron is Leaky-Integrate-and-Fire (LIF), which gradually decreases (or leaks) membrane potential at each timestep. The following equation describes the LIF mechanism of a neuron $i$:

$$U_i^t = \sum_{j \in N} W_{ij} S_j^{t-1} + \beta U_i^{t-1} - S_i^{t-1} U_{thr},$$

$$\text{where } \beta < 1 \text{ and } S_i^t = \begin{cases} 1 & \text{if } U_i^t > U_{thr}, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

in which, $S_i^t$ is the binary output of neuron $i$, and $U_i^t$ is its membrane potential at timestep $t$. While $\beta$ represents the leak factor by which the membrane potential is reduced at each timestep and $U_{thr}$ represents the membrane threshold. $N$ denotes the set of input neurons that is connected to $i$, and $W_{ji}$ denotes the synaptic weight of the $j \to i$ connection.

**Back-propagation method for model training**. Due to the time dimension encoding and the non-differentiable functions, the conventional ANN backpropagation is unsuitable for SNN training. According to Neftci et al. [6], the back-propagation of a spiking neuron begins by calculating $\Delta W_{ji}$, the gradient of the weight connecting neuron $i$ and $j$ accumulated over $T$ as follows:

$$\Delta W_{ji} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}}{\partial S_i^t} \frac{\partial S_i^t}{\partial U_i^t} \frac{\partial U_i^t}{\partial W_{ji}}, \quad (2)$$

where $\mathcal{L}$ is the loss function. Categorical cross-entropy is widely used in image classification. As $S_i^t$ is a thresholding function, computing $\Delta W_{ji}$ is intractable. To simplify this problem, the threshold function can be approximated with surrogate functions that is defined as follows:

$$\frac{\partial S_i^t}{\partial U_i^t} = \xi \max \left\{ 0, 1 - \left| \frac{U_i^t - U_{thr}}{U_{thr}} \right| \right\}, \quad (3)$$

where $\xi$ is a hyperparameter decay factor for back-propagated gradients [6], [7]. As gradients are accrued through time, $\xi$ is adjusted according to $T$. The larger the timesteps, the smaller the decay factor should be to avoid gradient exploding.

### B. Federated Learning

FL framework typically consists of one global aggregation server and a set of local training clients $\mathcal{C}$, in which each client $c$ possesses its private dataset $\mathcal{D}^{(c)}$. The server starts the training with a global initial model weight $\mathbf{W}_0$ and is distributed to the clients. For each round $r$, client $k$ receives the global updated model $\mathbf{W}_{r-1}$ aggregated from the previous round. Then, each client uses its private data samples for local training and obtains the locally updated model $\mathbf{W}_r^{(c)}$. The client transfers its updated parameters to the server for aggregation and produce the global model $\mathbf{W}_r$, the following formula describes this *FedAvg* algorithm [8]:

$$\mathbf{W}_i = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \mathbf{W}_i^{(c)}, \quad (4)$$

in which, $|\mathcal{C}|$ denotes the number of participated clients in the FL network. Note that there is a rich literature on alternatives for model aggregation algorithms. For simplicity, this work will adopt the *FedAvg* algorithm for evaluation without loss of generality. Fig. 4 illustrates an overview of FL with SNN integrated with the compression schemes that we are proposing in this work.

### C. SNN In-situ Training Hardware/System

This paper investigates on top of the client-server architecture in which the server or each client has SNN in-situ training capability. Many recent research works [9]–[11] focus on SNN in-situ training and have made significant progress. We expect that in-situ training will become an essential feature of neuromorphic hardware/systems in the near future.

## III. Robustness of SNN in FL with noisy communication

Recently, Patel et al. [12] investigated the impact of noisy input on the SNN training phase and showed that SNN models

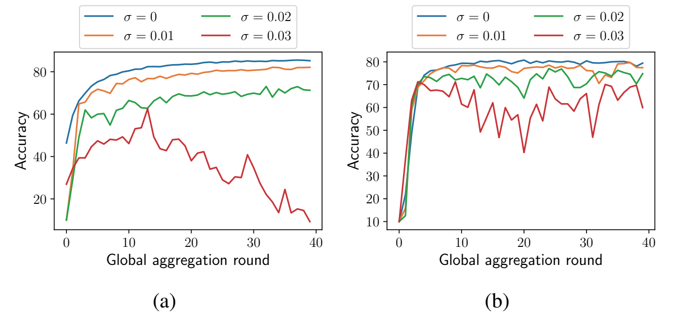


Fig. 2. The stability of a) ANN and b) SNN under various noise levels $\sigma$ using absolute values.

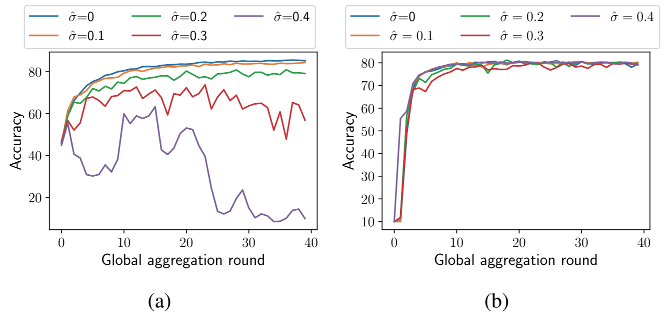


Fig. 3. The stability of a) ANN and b) SNN under various noise levels $\hat{\sigma}$ using relative percentage to the magnitude of the parameters.

can be more resilient than their ANN counterparts. This observation motivates us to explore the robustness of SNN to practical noisy communication in FL, which is an under-explored research area. In this section, we extend SNNs to the FL setting under noisy communication and compare the noise robustness of SNNs with ANNs.

**Setup**. To compare the noise robustness between SNNs and ANNs in a FL environment, we utilize VGG9 as the base architecture for both models, with stochastic gradient descent (SGD) optimizer and average pooling. We follow the setup in [4] and adopt similar training parameters. For SNN, we set the timesteps to 25, with a learning rate of 0.1 and momentum of 0.95, while for ANN, we set the learning rate of 0.001 and weight decay of $5e-4$. We set up the federated learning system with 5 clients, with a batch size of 32. We train for 40 global aggregation rounds. Before aggregating each global aggregation, each client trains on its private data for 5 local epochs. The model noise is added before transferring global parameters from server to client and locally trained parameters from client to server. The noise is generated following the Gaussian distribution $N(0, \sigma)$, in which the standard deviation, i.e., $\sigma$, indicates the strength.

**Noise generation**. In our evaluation, we vary the noise strength and compare the accuracy stability to verify the model robustness. We conduct experiments for ANN and SNN models by setting the noise strength. First, $\sigma$ is set as a fixed value from round to round, called absolute noise (Fig. 2). Second, $\sigma$ is fractionally dependent on the average magnitude of the transmitting parameters at each round, called relative noise (Fig. 3). With $\hat{\sigma}$ to annotate the relative noise strength, we have the following noise generation scheme:

$$\sigma_r = \hat{\sigma} \times \textsc{Avg}\left(\{|\mathbf{w}| : \forall \mathbf{w} \in \mathbf{W}_r\}\right), \tag{5}$$

in which $\sigma_r$ is the strength of the noise added to the parameters set, $\mathbf{W}_r$, when transmitting at round $r$.

**Robustness comparison for ANN and SNN**. In Fig. 2, three different absolute noise levels $\sigma \in \{0.01, 0.02, 0.03\}$ are compared with the no-noise baseline in both ANN (shown in Fig. 2(a)) and SNN (shown in Fig. 2(b)). According to our experimental results, even though absolute noise introduces instability to both models, the noise at $\sigma = 0.03$ causes the ANN model to be more saturated over time, while the SNN model is still robust enough to maintain relatively stable accuracy. On the other hand, in Fig. 3, we also compare the no-noise baseline of both models with four different relative noise levels $\hat{\sigma} \in \{0.1, 0.2, 0.3, 0.4\}$. As the relative noise strength increases, the accuracy of ANN drops dramatically and becomes unacceptable. However, the accuracy of SNN remains stable with all these noise levels. Therefore, we conclude that SNNs consistently outperform ANNs in terms of robustness in FL under noisy communication.

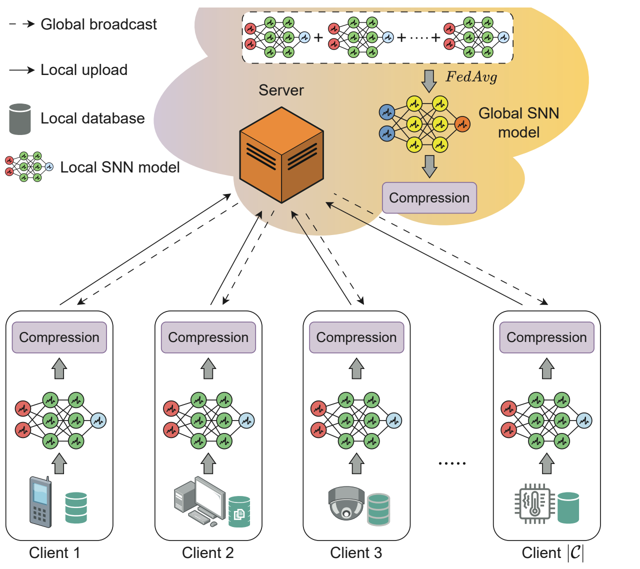


Fig. 4. Overview of Federated Learning with Spiking Neural Networks integrated with the proposed model compression algorithms.

## IV. Proposed Communication-efficient FL Algorithms with SNN

### A. Rationale

As explored in Section III, the property of robustness to noise demonstrated by SNN implies its potential resilience to other modes of manipulation of the parameters being transferred between FL clients and the server. In light of this observation, we are motivated to propose a FL algorithm with SNN that aims to reduce the communication load among the nodes while avoiding compromises in the quality of the

**Algorithm 1** FL with Top-$\kappa$ Sparsification (FLTS)

```
Input: $\mathcal{C}$, $\{\mathcal{D}^{(c)} : c \in \mathcal{C}\}$, $R$, $\kappa$
Output: $\mathbf{W}_R$
 1: Initialize $\mathbf{W}_0$
 2: $j \leftarrow 0$
 3: for $r = 1$ to $R$ do                              ▷ Global training loop
    Stage 1: Distribute global parameters
 4:     $\mathbf{W}'_{r-1} \leftarrow$ SPARSE$(\mathbf{W}_{r-1}, \mathbf{H}_{r-1}, \kappa)$, equation (6)
 5:     Distribute $\mathbf{W}'_{r-1}$ to clients
    Stage 2: Federated training (Client-side)
 6:     for each client $c \in \mathcal{C}$ do
 7:         Train model $\mathbf{W}_r^{(c)}$ with local dataset $\mathcal{D}^{(c)}$
 8:         $\mathbf{W}_r'^{(c)} \leftarrow$ SPARSE$(\mathbf{W}_r^{(c)}, \mathbf{H}_r^{(c)}, \kappa)$
 9:         Submit $\mathbf{W}_r'^{(c)}$ to the server
    Stage 3: Model aggregation
10:     $\mathbf{W}_r \leftarrow$ FEDAVG$\left(\left\{\mathbf{W}_r'^{(c)} : c \in \mathcal{C}\right\}\right)$
11: return $\mathbf{W}_R$
```

trained model. Fig. 4 shows the integration of the compression algorithms into the communication flow of FL with SNNs.

## B. Federated Learning with Top-$\kappa$ Sparsification

We propose Federated Learning with Top-$\kappa$ Sparsification (FLTS) leveraging the robustness of SNNs to improve communication efficiency. The details are described in Algorithm 1 in stages. At *Stage 1*, the updated model generated from the previous global aggregation is broadcast to all participating clients. At *Stage 2*, clients perform independent training by utilizing private data samples and submitting local gradients to the server. At *Stage 3*, the server performs the *FedAvg* algorithm to aggregate the data gathered from clients and generate a new global model. For the transmissions between clients and the server in *Stage 1* and *Stage 2*, we design function SPARSE to implement our Top-$\kappa$ Sparsification scheme as follows. The goal is to reduce the transferring load and, therefore, to lower the total bandwidth overhead in the FL process.

**Top-$\kappa$ Sparsification**. Our proposed Top-$\kappa$ parameters sparsification scheme is a compression protocol, which modifies the sparsification scheme in [13]. The goal is to transmit only top parameters with the largest gradients (absolute values) in the resulting model. The insight of this approach is that the magnitudes of parameter gradients are associated with their importance to the model update. Thus, we can potentially preserve the important information in communication by transferring only the model updates with large magnitudes. Specifically, assume that $\mathbf{W} \in \mathbb{R}^d$ is the trained model, and the corresponding gradient of this model is $\mathbf{H} \in \mathbb{R}^d$. Mathematically, the gradient of a client model after local training of round $r$ is $\mathbf{H}_r^{(c)} = \mathbf{W}_r^{(c)} - \mathbf{W}_{r-1}$. In the same manner, the gradient of the global model after aggregation is $\mathbf{H}_r = \mathbf{W}_r - \mathbf{W}_{r-1}$. In this work, we define a variable $\kappa \leq 1$ for the compression rate, i.e., the fraction of preserved parameters. Such that the number of parameters to be preserved is $u = \kappa|\mathbf{W}|$, in which $|\mathbf{W}|$ is the total number of parameters. The Top-$\kappa$ Sparsification function is defined as:

$$\textsc{Sparse}(\mathbf{W}, \mathbf{H}, \kappa) = \{\mathbf{w}_{i_1}, \mathbf{w}_{i_2}, ..., \mathbf{w}_{i_u}\}, \tag{6}$$

where $i_1, i_2, ..., i_u \leq |\mathbf{W}|$ are parameter indices and $\mathbf{h}_{i_1}, \mathbf{h}_{i_2}, ..., \mathbf{h}_{i_u} \in \mathbf{H}$ are gradients with top absolute values.

**Algorithm 2** FL with Dynamic-$\kappa$ Reduction (FLDR)

```
Input: $\mathcal{C}$, $\{\mathcal{D}^{(c)} : c \in \mathcal{C}\}$, $R$, $(\alpha, \omega)$
Output: $\mathbf{W}_R$
 1: Initialize $\mathbf{W}_0$
 2: $\kappa \leftarrow \alpha$                  ▷ Initialize $\kappa$ at the highest value
 3: for $r = 1$ to $R$ do
    Stages 1, 2 & 3: Execute as Algorithm 1 with current $\kappa$
    Stage 4 (new): Reduce $\kappa$ value
 4:     $\kappa \leftarrow$ REDUCE$(\kappa, \alpha, \omega, R)$, equation (7) or (8)
 5: return $\mathbf{W}_R$
```

## C. Federated Learning with Dynamic-$\kappa$ Reduction

Notice that for FLTS in the last section, the compression rate $\kappa$ is constant throughout the training process. However, as pointed out by Yan et al. in [14], [15], as the global model is gradually being developed through the global aggregation in FL training, the number of important parameters needed to be passed around can be gradually decreased. Specifically, we can potentially further reduce the communication cost by dynamically adjusting the compression rate $\kappa$. Therefore, in this section, we design a modified FLTS called Federated Learning with Dynamic-$\kappa$ Reduction (FLDR) to further reduce the total bandwidth consumption for FL with SNNs. The algorithm is presented in Algorithm. 2, where $\kappa$ is dynamically adjusted in each global aggregation round. We describe the heuristic $\kappa$ selection strategy as follows.

**Dynamic-$\kappa$ reduction**. The adjustment of $\kappa$ is implemented through the $\textsc{Reduce}(\kappa, \alpha, \omega, R)$ function. We implement two modes of $\kappa$ reduction. The first mode relies on linear reduction, in which the function is formulated as follows:

$$\textsc{Reduce}(\kappa, \alpha, \omega, R) = \kappa - \frac{\alpha - \omega}{R}. \tag{7}$$

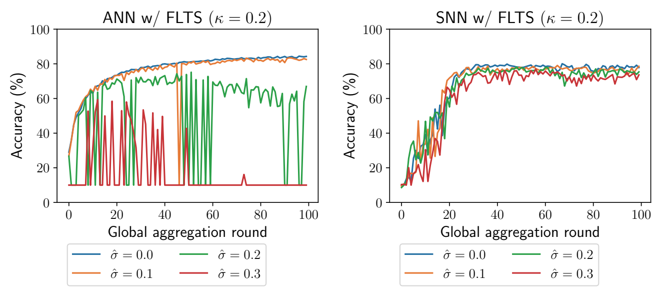
The second dynamic mode is exponential reduction and is formulated as follows:

$$\textsc{Reduce}(\kappa, \alpha, \omega, R) = \exp\left(\ln(\kappa) - \frac{\ln(\alpha) - \ln(\omega)}{R}\right). \tag{8}$$

In both functions, $\kappa$ is the compression rate of the previous round, $R$ is the number of global training rounds, $\alpha$ is the initial compression rate, and $\omega$ is the final compression rate. Through out $R$ rounds of global training, $\kappa$ is decreased slowly from $\alpha$ to $\omega$, and the $\textsc{Reduce}(\kappa, \alpha, \omega, R)$ function simply computes the next step of $\kappa$ reduction.

## V. Experimental Evaluation

In this section, the proposed FLTS and FLDR algorithms are evaluated. We utilize CIFAR10, a multi-class classification



(a) (b)

Fig. 5. Comparison of FL with Top-$\kappa$ Sparification under noisy communication between a) ANNs and b) SNNs.

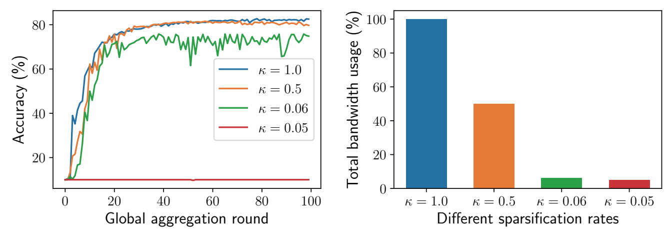
dataset popular for SNN performance evaluations, to evaluate our proposed compression approaches, including $50,000$ $32 \times 32$ RGB images of training data and $10,000$ images of validation data from the original dataset. The training data is equally split among clients. If not specified in the experiment, we adopt 5 clients in the evaluation, and all the clients participate in each global aggregation round. The VGG9 SNN model is applied to the FL framework for evaluation. The sequence length of SNNs, in which each neuron perceives the input data by receiving spikes, will be set to 25, leveraging the leaky-integrate-and-fire (LIF) model for neuron behavior. The training is conducted with SGD optimizer, and the learning rate is set to $0.1$ with a momentum of $0.95$. The batch size is configured to 32 across all devices. To better observe the training process, each client performs 1 local epoch per global round in our experiments, with a total of 100 global aggregation rounds. All the simulation, evaluations of the model and integrated compression algorithms performed on a centralized server with an 80 Core Intel(R) Xeon(R) CPU E5-2698 v4 @ 2.20GHz processor and 503GB of memory. They system also incorporates eight NVIDIA Tesla V100-SXM2-32GB, each with 32 GB of memory.

## A. FL with Top-$\kappa$ Sparsification under Noisy Communication

In this section, we conduct experiments on FL with Top-$\kappa$ sparsification (i.e., FLTS) under noisy communication. We aim to compare the performance of ANN and SNN in the setting aforementioned. The compression rate $\kappa$ is set to $0.2$ with various relative noise levels $\hat{\sigma} \in \{0.0, 0.1, 0.2, 0.3\}$. The results are illustrated in Fig. 5. We observe that noisy communication impacts the FL training for both ANN and SNN models. Nevertheless, SNN is significantly more robust than ANN. Specifically, as the noise level increases, the accuracy of ANN becomes much worse, or the model fails to converge, leaving very limited room for communication compression. This phenomenon validates that FL equipped with SNN allows much more bandwidth saving than ANN under practical noisy communication.

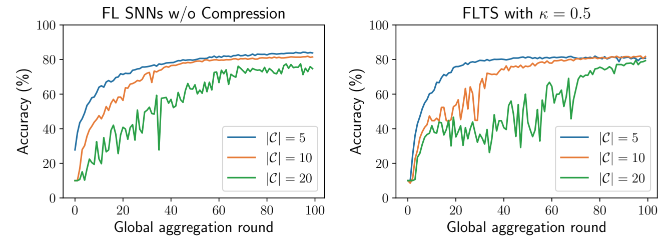
## B. Effect of Compression Rate Parameter $\kappa$ in FLTS

This section evaluates the impact of compression rate parameter $\kappa$. Fig. 6 presents the training results of FLTS under



(a) (b)

Fig. 6. The effectiveness of Top-$\kappa$ sparsification in SNNs by comparing the a) Accuracy, and b) Total bandwidth usage of different sparsification rate $\kappa$.



(a) (b)

Fig. 7. Impact of network size, i.e. different number of clients $|\mathcal{C}|$, on FL with SNNs a) Without compression, and b) FLTS with compression $\kappa = 0.5$.

various $\kappa$ settings such that $\kappa \in \{1.0, 0.5, 0.06, 0.05\}$. This evaluation aims to demonstrate the effectiveness of FLTS in terms of accuracy and total bandwidth utilization compared to the baseline without compression, i.e., $\kappa = 1.0$, for SNNs. We observe similar accuracy trends with compression rates of 50% ($\kappa = 0.5$) and the baseline without compression ($\kappa = 1.0$). The highest accuracy of $\kappa = 1.0$ is $82.7\%$ while $\kappa = 0.5$ is $81.7\%$. This result indicates that our FLTS algorithm has great potential to reduce bandwidth overhead in FL with SNNs. Fig. 6(a) also demonstrates that the FL algorithm normally performs even under $\kappa = 0.06$, with the highest accuracy of $75.8\%$, while the FL training does not learn anything under $\kappa = 0.05$. Our experimental results indicate that $6\%$ is the minimum compression rate of FLTS for VGG9 SNN with the CIFAR10 dataset.

## C. Impact of Network Size

In this section, we investigate the impact of network size. Fig. 7(a) and (b) display the impact of the number of clients ($|\mathcal{C}|$) on the accuracy of FL with SNNs without communication compression, i.e. $\kappa = 1.0$, and FLTS with compression rate $\kappa = 0.5$, respectively. First, as the number of clients increases, the accuracy curves degrade in uncompressed and compressed scenarios. This observation is expected as it takes more global aggregation rounds in FL to fuse the information in a network with a larger number of clients. In addition, our proposed compression method incurs higher variability in accuracy compared to the uncompressed baseline when the network size increases. Specifically, there are more fluctuations, especially in the early stages of the model training. We analyze that this is due to the fact that as the data are

TABLE I
BENCHMARKING DIFFERENT SETTINGS OF FLDR: 1) TOTAL BANDWIDTH CONSUMPTION (FRACTION COMPARED TO NO COMPRESSION) TO ATTAIN SPECIFIC LEVEL OF ACCURACY, AND 2) HIGHEST ACCURACY ACHIEVED

| **Metrics** | | **FLDR** | | | | | |
|---|---|---|---|---|---|---|---|
| | | **Linear Reduction FLDR-L** | | | **Exponential Reduction FLDR-E** | | |
| | | $\omega = 0.01$ | $\omega = 0.001$ | $\omega = 0.0001$ | $\omega = 0.01$ | $\omega = 0.001$ | $\omega = 0.0001$ |
| **Bandwidth for Accuracy** | 25% | 0.13 | 0.12 | **0.07** | 0.1 | 0.13 | 0.09 |
| | 40% | 0.11 | 0.1 | 0.11 | 0.1 | 0.1 | **0.08** |
| | 50% | 0.09 | 0.08 | 0.08 | 0.08 | 0.07 | **0.06** |
| | 60% | 0.08 | 0.09 | 0.08 | 0.07 | 0.07 | **0.06** |
| | 70% | 0.08 | 0.07 | 0.07 | 0.07 | 0.07 | **0.05** |
| | 75% | 0.13 | 0.1 | 0.1 | 0.09 | **0.06** | ▨ |
| **Highest Accuracy** | | 78.83% | 79.65 | 79.78% | 78.76% | 77.19% | 73.35% |

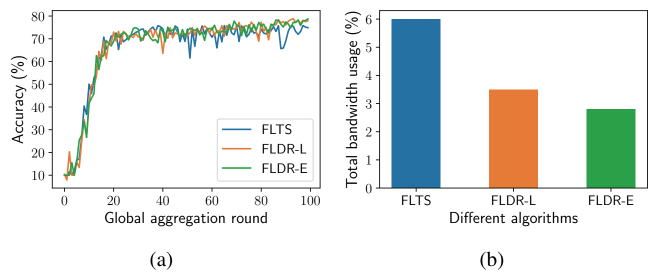


Fig. 8. Comparison of proposed algorithms: FLTS ($\kappa = 0.06$), FLDR-L ($\{\alpha = 0.06, \omega = 0.01\}$) and FLDR-E ($\{\alpha = 0.06, \omega = 0.01\}$).

spread to more clients, it becomes more difficult for the model to converge. Communication compression could further slow down the training process. Interestingly, we find that the final accuracy of our proposed FLTS algorithm is comparable to or even slightly higher than the baseline with no compression.

### D. Effect of the Final Compression Rate $\omega$ in FLDR

This section evaluates the effect of the final compression rate $\omega$ in FLDR. The highest accuracy is obtained by finding the maximum accuracy value within 100 global aggregation rounds. The motivation for this experiment is to understand the accuracy pattern when attempting to further reduce bandwidth usage by lowering $\omega$. As observed in TABLE I, we find no significant accuracy difference for linear reduction scheme under various $\omega \in \{0.01, 0.001, 0.0001\}$. Similarly, there is no substantial difference in total bandwidth consumption when varying the $\omega$ value. We conduct another experiment by varying $\omega$ in FLDR with exponential reduction. Different from FLDR-L, we observe that the accuracy trend of FLDR-E becomes more and more destabilized in exponential reduction as the $\omega$ value declines. The main reason for this observation is that the compression rate decline of the first few rounds in FLDR-E is much steeper compared to that of FLDR-L. Furthermore, we observe that the total bandwidth usage is reduced significantly in lower $\omega$ settings. These observations illustrate the capability of FLDR to maintain high model accuracy with significantly lower communication costs, making them promising solutions for FL scenarios where bandwidth availability is extremely limited.

### E. Comparison of FLTS and FLDR Algorithms

In this experiment, we compare our proposed FLTS and FLDR algorithms to investigate their characteristics further. For FLDR, two schemes are being evaluated that are based on equations (7) and (8), respectively. They are denoted as FLDR-L and FLDR-E. To ensure a fair comparison, $\kappa$ is set to 0.06 for FLTS, and we apply reduction parameters $\{\alpha = 0.06, \omega = 0.01\}$ for both FLDR-L and FLDR-E. As shown in Fig. 8(a), FLDR-L and FLDR-E are slightly better than FLTS, with accuracy values 78.83%, 78.76%, and 75.8%, respectively. Fig. 8(b) summarizes the total bandwidth consumption of three approaches, where FLDR-L (3.5%) and FLDR-E (2.79%) are also better than that of FLTS. Note that the model training fails to converge when we attempt to reduce the total bandwidth in FLTS to 5% in Fig. 6. However, Fig. 8 demonstrates that the communication efficiency with FLDR can be further improved in that the parameter reduction is doubled compared to the lowest threshold of FLTS.

## VI. RELATED WORK

**Robustness of SNN**. Patel et al. [12] investigate noise impacts on SNN models when noise is injected into the input and the training process. Kundu et al. [16] develop an SNN training algorithm that uses crafted input noise in order to harness the model's robustness to gradient-based attacks. Ma et al. [17] further extend this line of efforts by proposing a Noisy Spiking Neural Network (NSNN) paradigm, intentionally incorporating noise to improve the model's robustness against adversarial attacks and challenging perturbations. These studies reveal the effectiveness and resiliency of SNNs to noise added to stages of model training. However, they do not target the FL context as investigated in our work.

**FL with SNN**. Tumpa et al. [18] evaluate the performance of FL with SNN in heterogeneous systems. Following this line of research, recent work in [19]–[21] demonstrates the efficiency of SNN with various FL environments and applications. However, these works mainly focus on leveraging the energy efficiency of SNN in FL systems. Venkatesha et al. [4] demonstrate the robustness of SNN in FL applications, including straggling, model dropout, and gradient noise. In this work, we further examine the robustness of SNN and design FL sparsification algorithms for SNN training to reduce communication costs while maintaining model accuracy.

**Communication Compression in FL**. Various methods based on compressing the model updates have been proposed to overcome the insufficient bandwidth issues in FL systems [22], [23]. However, these works are based on the conventional ANN models. In contrast, several works have been conducted to explore the communication efficiency in FL systems with SNNs and provide insights into the trade-offs between communication load and accuracy [24]–[26]. However, they do not consider noisy communication in practical FL systems.

## VII. CONCLUSION

In this article, we presented a comprehensive study on the enhancement of communication efficiency in Federated Learning (FL) using Spiking Neural Networks (SNNs). In light of the inherent robustness of SNNs to noise, we designed Top-$\kappa$ Sparsification based schemes to reduce communication cost in FL without compromising model accuracy. The empirical results demonstrated that FL with SNNs saves significantly more bandwidth than their ANNs counterparts under noisy communication. Specifically, under the effects of noise and compression in communication, SNNs can still achieve high model accuracy, while the training for ANNs may fail to converge. Our research findings lay the foundation for further exploring the characteristics of SNNs as an alternative to ANNs to improve network efficiency in FL.

## VIII. Acknowledgments

This article has been authored by an employee of National Technology & Engineering Solutions of Sandia, LLC under Contract No. DE-NA0003525 with the U.S. Department of Energy (DOE). The employee owns all right, title and interest in and to the article and is solely responsible for its contents. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this article or allow others to do so, for United States Government purposes. The DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan https://www.energy.gov/downloads/doe-public-access-plan.

## References

[1] M. Pesce, "Ai has a terrible energy problem. it's about to hit crisis point," 2024, accessed on April 30, 2024. [Online]. Available: https://cosmosmagazine.com/technology/ai-copilot-chat-gpt-energy/

[2] M. Dampfhoffer, T. Mesquida, A. Valentian, and L. Anghel, "Are snns really more energy-efficient than anns? an in-depth hardware-aware study," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 7, no. 3, pp. 731–741, 2022.

[3] S. Kundu, R.-J. Zhu, A. Jaiswal, and P. A. Beerel, "Recent advances in scalable energy-efficient and trustworthy spiking neural networks: from algorithms to technology," in *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 13 256–13 260.

[4] Y. Venkatesha, Y. Kim, L. Tassiulas, and P. Panda, "Federated learning with spiking neural networks," *IEEE Transactions on Signal Processing*, vol. 69, pp. 6183–6194, 2021.

[5] H. Ye, L. Liang, and G. Y. Li, "Decentralized federated learning with unreliable communications," *IEEE Journal of Selected Topics in Signal Processing*, vol. 16, no. 3, pp. 487–500, 2022.

[6] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 51–63, 2019.

[7] S. M. Bohte, "Error-backpropagation in networks of fractionally predictive spiking neurons," in *International conference on artificial neural networks*. Springer, 2011, pp. 60–68.

[8] 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.

[9] J. Li, H. Xu, S.-Y. Sun, N. Li, Q. Li, Z. Li, and H. Liu, "In situ learning in hardware compatible multilayer memristive spiking neural network," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 14, no. 2, pp. 448–461, 2022.

[10] L. Ma, G. Wang, S. Wang, and D. Chen, "Simulation of in-situ training in spike neural network based on non-ideal memristors," *IEEE Journal of the Electron Devices Society*, vol. 11, pp. 497–502, 2023.

[11] S. K. Vohra, S. A. Thomas, M. Sakare, and D. M. Das, "Circuit implementation of on-chip trainable spiking neural network using cmos based memristive stdp synapses and lif neurons," *Integration*, vol. 95, p. 102122, 2024.

[12] K. P. Patel and C. D. Schuman, "Impact of noisy input on evolved spiking neural networks for neuromorphic systems," in *Neuro-Inspired Computational Elements Conference*. ACM, 2023, pp. 52–56. [Online]. Available: https://dl.acm.org/doi/10.1145/3584954.3584969

[13] F. Zheng, C. Chen, L. Lyu, and B. Yao, "Reducing communication for split learning by randomized top-k sparsification," *arXiv preprint arXiv:2305.18469*, 2023.

[14] G. Yan, H. Wang, and J. Li, "Seizing critical learning periods in federated learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 8, 2022, pp. 8788–8796.

[15] G. Yan, H. Wang, X. Yuan, and J. Li, "Criticalfl: A critical learning periods augmented client selection framework for efficient federated learning," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 2898–2907.

[16] S. Kundu, M. Pedram, and P. A. Beerel, "Hire-snn: Harnessing the inherent robustness of energy-efficient deep spiking neural networks by training with crafted input noise," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. Los Alamitos, CA, USA: IEEE Computer Society, oct 2021, pp. 5189–5198. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/ICCV48922.2021.00516

[17] G. Ma, R. Yan, and H. Tang, "Exploiting noise as a resource for computation and learning in spiking neural networks," *Patterns*, vol. 4, no. 10, p. 100831, 2023. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2666389923002003

[18] S. A. Tumpa, S. Singh, M. F. F. Khan, M. T. Kandemir, V. Narayanan, and C. R. Das, "Federated learning with spiking neural networks in heterogeneous systems," in *2023 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2023, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/10238618/

[19] K. Xie, Z. Zhang, B. Li, J. Kang, D. Niyato, S. Xie, and Y. Wu, "Efficient federated learning with spike neural networks for traffic sign recognition," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 9, pp. 9980–9992, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9784851/

[20] O. Aouedi, K. Piamrat, and M. Sûdholt, "HFedSNN: Efficient hierarchical federated learning using spiking neural networks," in *Proceedings of the Int'l ACM Symposium on Mobility Management and Wireless Access*. ACM, 2023, pp. 53–60. [Online]. Available: https://dl.acm.org/doi/10.1145/3616390.3618288

[21] Y. Liu, Z. Qin, and G. Y. Li, "Energy-efficient distributed spiking neural network for wireless edge intelligence," *IEEE Transactions on Wireless Communications*, pp. 1–1, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10472876/

[22] H. Wang, S. Sievert, S. Liu, Z. Charles, D. Papailiopoulos, and S. Wright, "Atomo: Communication-efficient learning via atomic sparsification," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2018/file/33b3214d792caf311e1f00fd22b392c5-Paper.pdf

[23] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Sparse binary compression: Towards distributed deep learning with minimal communication," in *2019 International Joint Conference on Neural Networks (IJCNN)*, 2019, pp. 1–8.

[24] N. Skatchkovsky, H. Jang, and O. Simeone, "Federated neuromorphic learning of spiking neural networks for low-power edge intelligence," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 8524–8528. [Online]. Available: https://ieeexplore.ieee.org/document/9053861/

[25] S. Chaki, D. Weinberg, and A. Özcelikkale, "Communication trade-offs in federated learning of spiking neural networks." [Online]. Available: http://arxiv.org/abs/2303.00928

[26] Z. Liu, Q. Zhan, X. Xie, B. Wang, and G. Liu, "Federal SNN distillation: A low-communication-cost federated learning framework for spiking neural networks," in *Journal of Physics: Conference Series*, vol. 2216, no. 1, 2022, p. 012078. [Online]. Available: https://iopscience.iop.org/article/10.1088/1742-6596/2216/1/012078