Federated Learning with Spiking Neural Networks in Heterogeneous Systems

Sadia Anjum Tumpa, Sonali Singh, Md Fahim Faysal Khan,
Mahmut Tylan Kandemir, Vijaykrishnan Narayanan, Chita R. Das
School of Electrical Engineering and Computer Science, *Pennsylvania State University*
University Park, PA, USA
sbt5360@psu.edu, singhsonali0503@gmail.com, mzk591@psu.edu, mtk2@psu.edu, vxn9@psu.edu, cxd12@psu.edu

2023 IEEE Computer Society Annual Symposium on VLSI (ISVLSI) | 979-8-3503-2769-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/ISVLSI59464.2023.10238618

***Abstract*—With the advances in IoT and edge-computing, Federated Learning is ever more popular as it offers data privacy. Low-power spiking neural networks (SNN) are ideal candidates for local nodes in such federated setup. Most prior works assume that the participating nodes have uniform compute resources, which may not be practical. In this work, we propose a federated SNN learning framework for a realistic heterogeneous environment, consisting of nodes with diverse memory-compute capabilities through activation-checkpointing and time-skipping that offers $\sim$4$\times$ reduction in effective memory requirement for low-memory nodes while improving the accuracy upto 10% for non-independent and identically-distributed data.**

***Index Terms*—Federated Learning, Neuromorphic Computing, Spiking Neural Network (SNN), Heterogeneous Systems, Internet of Things**

I. Introduction

Spiking neural networks (SNNs), a class of brain-inspired neuromorphic computing algorithms, are becoming increasingly popular for their low-latency and energy-efficient signal-processing ability. Their stateful, temporal, and event-driven processing paradigm on binary inputs or 'spikes' and their locally-dense, globally-sparse connectivity set them apart from the mainstream dense, high-precision and often stateless deep/analog neural networks (ANNs). Custom hardware implementing SNNs are particularly well-suited for processing the output of neuromorphic sensors [1] of different modalities (e.g. visual, auditory, tactile etc.) [2] close to the sensor facilitating on-chip learning. Having inspired a plethora of research in academia [3] and industry [4], [5] alike, SNNs are gradually finding their way into commercial products [6], for various applications.

Concurrently, the growing demand for ubiquitous intelligence, combined with the need to keep data secure and private, has precipitated the concept of Federated Learning (FL [7]), a distributed and privacy-preserving learning approach among edge devices (also referred to as clients or nodes in this text) and a cloud server. In FL, the model broadcast by the central server is trained in a 'distributed fashion' on the 'edge nodes' which instead of sharing local data, periodically convey the model updates to the central server, that in turn aggregates these updates into a global model and transmits it back to the edge nodes for further training providing the advantage of collaborative learning and is gradually being deployed in a variety of applications [8]. In a practical FL scenario of the current and future generations, there can be a wide variety in the type of devices or sensor networks that collaboratively contribute towards updating a single global model. For example, a personalized intelligent health monitoring system may involve an activity tracker, a smart-phone, a laptop computer and a virtual assistant such as Amazon Alexa [9] when training a neural network on locally-available data and then periodically uploading their local models for a global aggregation. In such a heterogeneous system, it is natural for the edge devices to have variable underlying technologies, form factors, thermal design powers (TDPs) and consequently, differing memory and compute capabilities.

This work is supported in part by National Science Foundation (NSF) grant nos. 1955815, 1822923 and Center for Brain-inspired Computing (C-BRIC).

Recent works [10] demonstrate the training of large-scale SNNs through backpropagation-through-time (BPTT) [11] in a federated setup. However, the SNN-BPTT training can be extremely memory and compute-intensive [12], [13], and in fact its memory requirement scales as $O(T)$, where T is the number of SNN timesteps and is often very large. As a result, in a real-world FL scenario consisting of a variety of heterogeneous client nodes (refer to Figure 1), *not* all clients may be able to participate in the global model updates due to power and consequently memory resource constraints. This can lead to adverse effects such as slowing down the training, or limiting the robustness or applicability of the global model, while also under-utilizing the plentiful of data available locally. Although prior work [13] attempts to reduce the memory consumption of BPTT-based SNN training by leveraging activation checkpointing in the time domain along with time-skipping, it only *targets a single node*. In contrast, in this work, we investigate FL in SNNs while accounting for the 'resource-constraint-induced heterogeneity' of the clients in a *distributed setup* which is a more challenging problem from a convergence and scaling perspective. Specifically, we explore the following fundamental questions: *Can we enable the memory-limited nodes to participate in training and evolving the global model?* If so, *would such a heterogeneous federated learning environment perform as well as its homogeneous counterpart?*

To tackle the first question, we propose a novel approach to enable the SNNs to learn in a heterogeneous memory environment, which involves mapping the basic fully-unrolled SNN to powerful clients with a large memory capacity, whereas running vanilla activation-checkpointing [13] and *skipper* [13] SNNs on clients with moderate and low available memory capacities, respectively. Through extensive and structured experimentation on popular SNNs and datasets, we successfully demonstrate that SNNs in a client memory-induced heterogeneous FL environment can not only learn, but also perform on par with homogeneous setups explored in prior work [10]. Further, we demonstrate that nodes with low memory capacity, that would otherwise have low network performance can be supported by 'memory-rich' nodes to learn and achieve similar accuracy to a 'memory-rich' homogeneous environment for non-independent and identically distributed (non-IID) data. Our contributions in this work can be summarized as follows:

- To the best of our knowledge, we are the first work to investigate memory resource heterogeneity in the context of SNNs in a federated setting, which forms a key component in the overall power consumption of the client devices.
- We propose a novel approach to enable SNN training in a heterogeneous FL environment, without making any invasive changes to the FL framework and with a negligible overhead. Specifically, the memory overhead is $\sim$4$\times$ lower than the unrolled SNN from [10].
- We report competitive performances compared to the homogeneous setup on VGG9 and LeNet networks as well as CIFAR10 and DVS-gesture datasets, indicating that our approach is applicable to

979-8-3503-2769-4/23/$31.00 ©2023 IEEE

both frame and event-based data.

- We show how memory-constrained nodes can competitively participate in global model updates with some support from 'memory-rich' nodes and improve accuracy by up to 10% for non-IID data.
- Last but not the least, we experiment with two different global aggregation techniques (viz. plain and weighted averaging), and observe that both lead to similar accuracies.

II. Related Work

In this section, we summarize the existing works on federated SNN training and those on heterogeneous systems in federated framework that inspired our current work on heterogeneous federated SNN. Most existing SNN works investigate the energy-efficiency of SNNs with entire data at one place. With the advances of edge computing, researchers are looking into how energy-efficiency of SNN can be leveraged in distributed systems. Keeping that in mind [14] explores collaborative federated learning for on-device SNNs by generalizing the single-SNN learning rule in [15] for probabilistic SNNs. Another work [16] uses an approximate back-propagation technique as proposed in [17], [18] to train SNNs in a leader based federated neuromorphic learning framework (LFNL). Recently, for low latency and scalable SNN training in federated setup, [10] uses BPTT based SNN training technique along with surrogate gradient method that was proposed in [19] which we adopted as our baseline federated set up assuming that all the edge devices have enough memory capacity to train the SNN model locally. However, real world systems are inherently heterogeneous in resource availability. Several works have been looking into the heterogeneous resources aspects of federated learning. Nishio et al. have explored a client selection FL protocol that actively manages clients with resource constraints allowing the server to aggregate as many model updates as possible from clients [20]. Another work [21] also demonstrated strategies to enhance FL training to address heterogeneous computation and communication capabilities by reducing the number of local parameters at the same time generating a stable global model update. However, these heterogeneous works have used traditional deep neural networks for training purpose. In contrast, this work leverages the energy-efficiency and robustness of SNNs to address the problem of non-uniform compute/memory capabilities of the participating nodes in a federated learning environment.

III. Background

In this section, we provide the necessary background on federated learning, spiking neural networks and their training, and also discuss the prior work that enables memory-constrained training of an SNN.

A. Federated Learning

Federated Learning (FL) [7] refers to a distributed training methodology where a high-quality shared global model is learned from a vast number of local models running on different types of edge devices while preserving local data privacy and reducing the communication cost. Moreover, only the gradients of randomly selected local models, also referred to as clients, are sent to the central server where the global model update is done. The central server then pushes the global model to the local nodes. Generally speaking, in a federated setup, the optimization goal is:

$$\min_{\boldsymbol{W}} F(\boldsymbol{W}) := \sum_{i=1}^{N} \{f_i(\boldsymbol{W}) := \mathbb{E}_{x\sim\mathcal{P}_i}[\ell(\boldsymbol{W};x)]\}, \tag{1}$$

where N is number of clients each having $\mathcal{P}_i$ data distribution locally, $f_i(\boldsymbol{W})$ is the local model at edge node i having parameters $\boldsymbol{w}$ and the loss function ℓ, and $\mathbb{E}$ denotes the expected value of the loss across multiple training iterations. FL involves learning across many local nodes and in reality, these nodes have varying memory and compute capabilities, which makes the system 'inherently heterogeneous'. Since many of these edge nodes run on very low power and have limited memory and compute resources, energy-efficient models such as those based on SNNs [22] are of great interest. Although, FL with SNNs and traditional ANNs is similar in that both involve training a model on data from multiple devices, with the goal of minimizing a loss function over the combined data, there are important differences in the nature of the data, the non-differentiability of spiking neurons, the complexity of SNN dynamics, and the sparsity of SNNs that can affect the design of FL systems for SNNs.

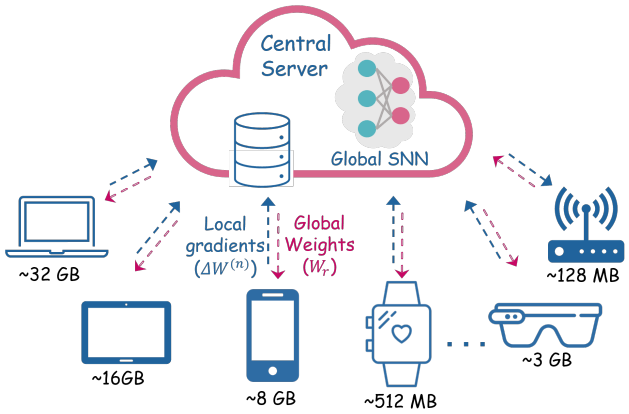

Fig. 1: Federated SNN in a Heterogeneous System

B. Spiking Neural Networks

SNNs are biologically-inspired neural networks that transmit information as discrete binary events or spikes. Once the internal state of a spiking neuron reaches a pre-defined threshold, it 'fires' an output spike and resets its state to a resting potential. In this work we consider the popular leaky-integrate-and-fire (LIF) neuron model [23] for its simplicity and scalability, while noting that our proposed techniques are agnostic to this choice. For digital simulations, a discrete-time formulation of the LIF neuron is as follows:

$$U_t^l = \lambda U_{t-1}^l + W^l o_t^{l-1} - \theta o_{t-1}^l, o_{t-1}^l = \begin{cases} 1, & \text{if } U_{t-1}^l > \theta \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where U_t^l is the neuron's membrane potential (internal neuron state) at time t, l is the layer index, λ (≤ 1) is the membrane potential leak, W_l is the weight matrix connecting layers $l-1$ and l, o_t is the spike vector at time t, and θ is the firing threshold potential. The first term on the right hand side of Eq. (2) carries forward the neuron state from the previous time-step to the current time-step (modulated by leak λ); the second term is a weighted sum of the spikes coming from the previous layer and the third term arises from the thresholding non-linearity that decreases the membrane potential by θ if an output spike o_{t-1}^l is generated by a neuron (2nd part of Eq. 2).

Backpropagation-through-time (BPTT) [11] combined with surrogate gradients has emerged as an effective technique to train deep SNNs directly [12]. SNN-BPTT training also enables learning with event-based data, which are often present in multiple modalities in nature being more inclusive of real-world scenarios. BPTT in SNNs entails minimizing a global loss function L by applying gradient descent that propagates *backwards* in time. As a result, the gradient descent weight update δW for layer l is governed by:

$$\begin{aligned} \delta W^l &= -\eta \frac{\partial L_t}{\partial W^l} = -\eta \sum_{s=0}^{t} \mu_s^l {o_s^{l-1}}^\top, \\ \text{s.t. } \frac{\partial L_t}{\partial U_s^l} &= \mu_s^l = \text{diag}(\sigma'(U_s^l)) {W^{l+1}}^\top \mu_{s+1}^{l+1} + \lambda \mu_{s+1}^l \end{aligned} \tag{3}$$

where σ' is the smooth surrogate function for computing the gradient of the activation function [12].

To reduce the high memory footprint of SNN training caused by temporal unrolling, [13] *skipper* saves or checkpoints only a subset of the total activations C in time T and selectively 'skips' or re-computes these activations based on time-skipping fraction p. In this work, we leverage these knobs to enable FL in SNNs in a heterogeneous FL environment as explained in the next section.

IV. Federated SNN Training on Heterogeneous Clients

We first establish the baseline (homogeneous) federated training framework in SNN from [10], and then build our heterogeneous training methodology on top of it. The federated framework consists of a central base station with N clients, each client $n \in \{1, ..., N\}$ having its own private dataset $D^{(n)}$ stored locally. The base station initially disseminates a randomly initialized untrained model W to all of its clients, along with the number of timesteps T, for which the SNN needs to be trained. Assuming that the clients have sufficient memory for training with a given T, each of the clients trains their local model $W^{(n)}$ natively for K epochs with an initial learning rate $\eta^{(n)}$, which leads to an aggregated model update $\Delta W^{(n)} = \sum_1^K \eta^{(n)} \delta W^{(n)}$ at the client, where δW are the SNN weight gradients as per Equation (3). A series of K local epochs is called a *round* r. In every round r, a randomly-chosen subset P (called participating clients) of the total N clients upload their aggregated model updates $\Delta W^{(n)}$ to the base station, which in turn, updates the global model as per the $FedAvg$ algorithm [7]:

$$W_r = W_{r-1} + \frac{1}{\sum_{n \in P} |D_r^{(n)}|} \sum_{n \in P} |D_r^{(n)}| \Delta W_r^{(n)}. \quad (4)$$

This global model is broadcast again to the participating clients for the next round and thus the learning progresses. This global aggregation scheme incorporates a weighted-averaging based on the number of training samples available at each node to accommodate the differences in learning volume of training data available at different nodes. Additionally, a training-type based weighted averaging technique (*wFedAvg*) for global aggregation has also been explored in our work introducing weight-proportion hyperparameter represented as $\omega^{(n)}$ for n-th participating client such that $\sum_n \omega^{(n)} = 1$. When models with same training-type are aggregated to the global model W_r in any round r, equal weighing of the local model updates is used similar to *FedAvg*, but, when models with different training-type aggregate to the global model, we choose higher value of $\omega^{(n)}$ for more accurate models (i.e., Basic and Checkpointed) and lower value of $\omega^{(n)}$ for less accurate model (Skipper) in this later approach. The use of advanced aggregation methods for federated frameworks (e.g., FedProx [24] and FedMA [25]) should further augment the value of our approach. However, that is orthogonal to the primary focus of our current work and will be explored in future work.

In a practical heterogeneous federated training framework, the clients often have different memory capabilities, as shown in Figure 1. Let $M_a^{(n)}$ denote the memory availability of client n, and let M_{req} denote the memory required to train an SNN for a given timestep, T. Thus, client n will only be able to train the SNN on its local data if $M_{req} < M_a^{(n)}$, which may not be the case for many low-power, resource-constrained devices.

Note however that, we can still enable these memory-constrained nodes to participate in the global model updates by applying a novel and effective technique, in which, the key idea is to *run either the original (basic, time-unrolled version), or the activation-checkpointed or the activation-checkpointed-with-time-skipped (skipper) version of SNN-BPTT on a client device depending on its available memory*. We know from [13] that, $M_{req}(original) > M_{req}(checkpointed) > M_{req}(skipper)$ and we leverage this characteristic to enable federated SNN learning on *memory-rich*, *memory-limited* as well as *memory-deficient* nodes, respectively. In this context, we define *memory-rich* clients which have enough memory available to run the temporally unrolled SNN-BPTT. In comparison, the *memory-limited* clients are the ones that do not have enough memory to run the unrolled version, but can accommodate the checkpointed SNN with C checkpoints, C being the number of times the activations are saved, and finally, the *memory-deficient* clients are those that cannot accommodate either, and therefore, must resort to time-skipping with activation-checkpointing with C checkpoints and p which indicates fraction of timesteps that are to be skipped to participate in the learning. A practical real-world federated learning scenario could consist of any proportion of these three types of client devices and each participating client trains a local copy of the model using either the original, checkpointed, or skipping-based SNN depending on its memory availability. The weight updates emanating from each of these model types are of exactly the same size since the checkpointing and time-skipping only happen in the 'temporal dimension', and can effectively contribute to the model updates at the base station.

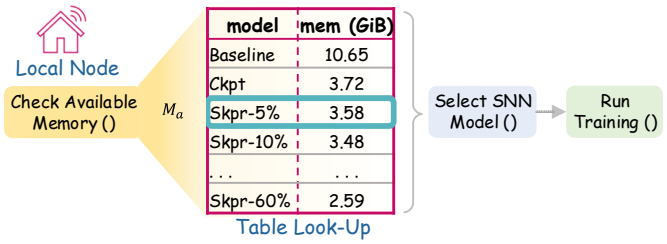

Fig. 2: Model selection methodology at the heterogeneous local nodes

The next question then arises: *how would the participating nodes decide which type of SNN training is best suitable for them?* Since the base station is not aware of the device configurations of the client nodes and the latter are unaware of the intricacies of the downloaded SNN model, the base station can use some extra 'metadata' to convey this information. We solve this with the help of a tiny *look-up table*, which the base station broadcasts to the clients only once at the beginning of the training. From the look up table consisting of the required memory to train the different SNN versions as shown in Figure 2, a participating client can decide which type of SNN model it can train for effective contribution to the global model with respect to its available memory.

V. Evaluation

To evaluate the performance of our method, we consider the scenario of 10/2 federated learning with $N = 10$ total clients and $P = 2$ participating clients at each round for experiments on two publicly available datasets namely CIFAR10 and neuromorphic DVS-Gesture. The VGG9 and LeNet SNNs chosen for our study are popular and standard SNN workloads evaluated in several prior work [5], [10], [13], [19], [26]. For our experiments, we extended the codebase of [10] to support activation-checkpointing and *skipper* in the federated framework and verified its functionality. The training hyperparameters are listed in I and rest are adopted as described in [10] and these were kept constant across the memory-constrained SNN variants for a particular network and dataset combination. We used initial learning rate of 0.1 and dynamically reduced it by 5 if the testing accuracy does not improve for 5 consecutive rounds. All our experiments are done using Pytorch 1.9.1 and executed on servers with NVIDIA's Titan RTX, Titan V and A100 GPUs. For reporting the memory consumption of all models, we use the NVIDIA Titan RTX measurements for consistency.

TABLE I: Mem. required for SNN training in a 10/2 (homogeneous) federated setup. Here, C= #checkpoints, p=time-skipping fraction

	VGG9, T = 50, Batch Size = 64 (CIFAR10)							**LeNet**, T = 400, Batch Size = 32 (DVS-Gesture)					
Train Type	Basic	C=5	C=5, p=5%	C=5, p=10%	C=5, p=20%	C=5, p=40%	C=5, p=60%	Basic	C=4	C=4, p=20%	C=4, p=40%	C=4, p=60%	C=4, p=80%
Mem.(GiB)	10.65	3.72	3.58	3.48	3.37	3.02	2.59	11.26	3.95	3.49	3.02	2.53	2.04

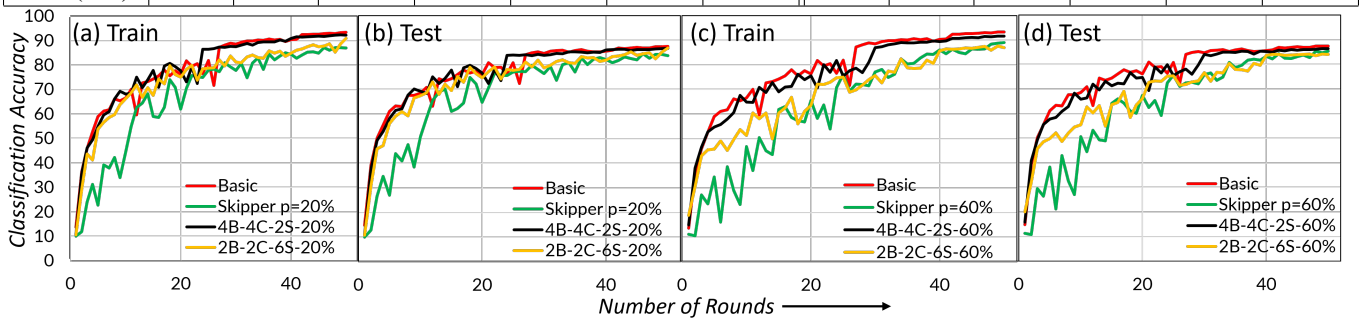

Fig. 3: Convergence plot for 10/2 *federated heterogeneous-clients* set-up with VGG9 on CIFAR10 *IID* data. p = time-skipping fraction, $xB - yC - zS - w\%$ represents x,y & z nodes running on Basic, Checkpointed and Skipper SNN with $w\%$ time-skipping respectively.

VI. Experiments

In this section, we conduct a rich set of experiments to study the impact of different types of 'memory-induced heterogeneity' in the clients of the federation. Specifically, we consider two types of heterogeneous FL setups in SNNs: (i) **Mixed-memory setup** which consists of a mix of *memory-rich*, *memory-limited*, and *memory-deficient* clients, and (ii) **Memory-constrained setup** consisting of only *memory-deficient* clients. To demonstrate the efficacy of our heterogeneous federated SNN learning framework, we compare it against a 'homogeneous' baseline in which all clients run the **same** type of model. We consider 3 types of the homogeneous federated setup: (a) **All clients are *memory-rich*** and run the same basic fully unrolled SNN model as per [10]. (b) **All clients are *memory-limited*** and run the same checkpointed network. (c) **All clients are *memory-deficient*** and run the same *skipper* network with constant p. For the above experiments, we assume that the local datasets available at the individual nodes are IID (i.e., independent and identically distributed). Lastly, we forgo this assumption and train the heterogeneous federated SNN framework with non-IID data, thus simulating a highly realistic and challenging platform representative of real-world scenarios and demonstrate the robustness of our approach.

A. Heterogeneous FL with Mixed Memory Clients

In this experimental setup, the clients have differing memory capacities available. The rationale behind this study is to simulate a real-world scenario, where the edge devices have variable computational capabilities and accompanying TDPs, for e.g., a laptop computer, a smart-phone, a VR headset etc. (refer to Figure 1). In such a heterogeneous FL environment, the *memory-rich* nodes (e.g., laptop computer) can run the original fully unrolled SNN, the *memory-limited* nodes (e.g., smart-phone) can train the checkpointed SNN whereas the *memory-deficient* node (e.g., VR headset) can run a *skipper* SNN model with an appropriate time-skipping fraction p.

Figure 3 shows the convergence plots for such a scenario. In each of the plots we show the classification accuracy vs. the number of federated training rounds by varying the proportion of *memory-rich*, *memory-limited* and *memory-deficient* nodes and compare it with the corresponding homogeneous setup. For example, Figure 3 shows the convergence curves for the unrolled (Basic) SNNs running on homogeneous *memory-rich* nodes, *skipper* with $p = 20\%$ and 60% running on homogeneous *memory-deficient* nodes and two flavors of heterogeneous mixed memory nodes, where xB-yC-zS-$w\%$ denotes x basic (B) or fully unrolled SNNs, y checkpointed (C) SNNs with $C = 5$, and z *skipper* (S) SNNs with $p = w\%$. We observe that all the heterogeneous federated setup variants are able to converge on the training set and generalize well on the held-out validation set. Further, when the proportion of *memory-deficient* nodes (running *skipper*) is relatively small in the federation, the heterogeneous federated setup converges as quickly as its homogeneous counterpart (4B-4C-2S-20% vs Basic and 4B-4C-2S-60% vs Basic). When the *memory-deficient* nodes are in a majority (2B-2C-6S-20% and 2B-2C-6S-60%), the networks still converge with only a slightly lower performance, even with aggressive time-skipping ($p = 60\%$). Another observation is that the *memory-deficient* homogeneous setup, where all nodes run *skipper*, ($p = 20\%$ or 60%) are much slower to converge compared to any of the heterogeneous nodes. This leads us to the key insight that in a heterogeneous SNN FL setup, the weaker nodes (limited by memory and ultimately due to power) can still perform on par with *memory-rich* homogeneous nodes when supported by *memory-rich* nodes from time-to-time. We make a similar set of observations for the LeNet SNN trained on the DVS-gesture dataset as well, whose convergence graphs are not shown in the interest of space.

Figures 4(a) and (c) report the final test accuracy of VGG9 and LeNet SNNs in different mixed-memory heterogeneous FL scenarios and compare it with that of homogeneous settings viz. unrolled SNN (Basic), checkpointed (C=5 for VGG9, C=4 for LeNet) and homogeneous *skipper* (case (c) as discussed in section VI). We notice that the heterogeneous variants closely match the accuracy of the Basic and checkpointed SNN. When compared to the homogeneous *memory-deficient* variants, the heterogeneous framework outperforms the former when p is low or the number of *memory-deficient* nodes are few. However, this trend tends to reverse for aggressive time-skipping (large p) or for larger proportion of *memory-deficient* nodes. Table I reports the memory requirements for training all the different variants of VGG9 SNN on CIFAR10 and LeNet SNN on DVS-gesture as measured on the NVIDIA's Titan RTX. The original unrolled SNN has a large memory footprint, that is considerably lowered by the checkpointed SNN which can be lowered even further with the help of *skipper* SNNs, in proportion to time-skipping percentage p.

B. Heterogeneous FL with Memory-Deficient Clients

In these experiments, all the clients have differing available memory capacities, with the added constraint that they only have enough memory to accommodate *skipper* SNNs with varying time-skipping fractions p. With such an experimental setup, our aim is to model a real-world heterogeneous FL scenario where all the devices are power- and consequently resource-constrained. As an example, a set

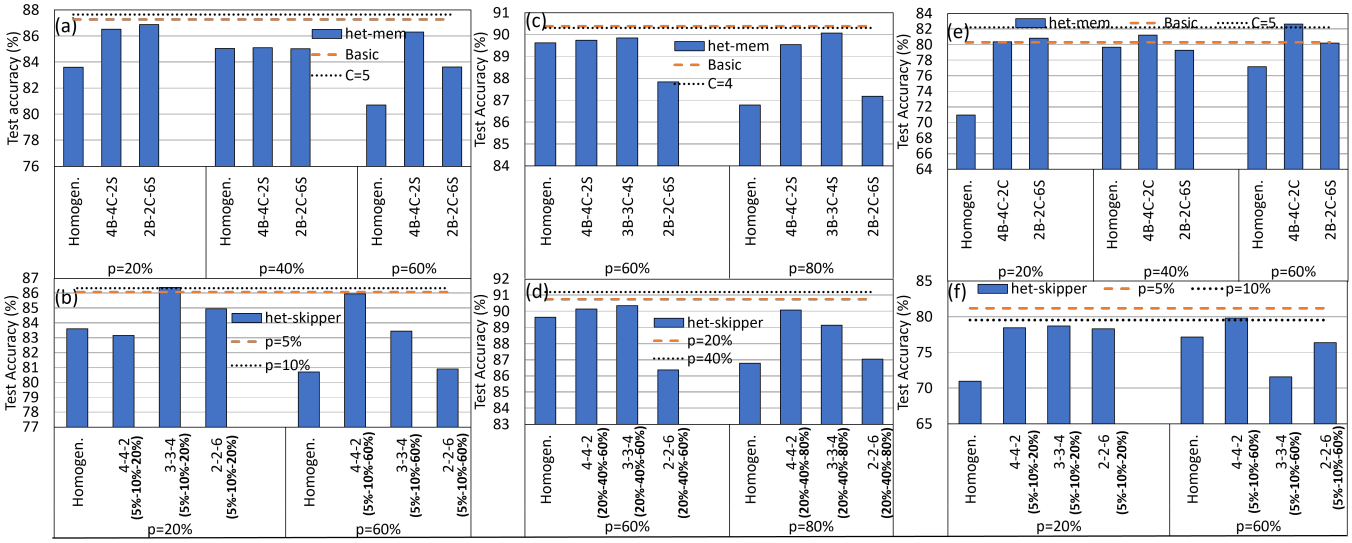

Fig. 4: Test accuracy for different types of heterogeneous-clients set-up in 10/2 federated learning. C= Number of Checkpoints, p=time-skipping fraction. $xB-yC-zS$ represents x,y & z nodes running on Basic, Checkpointed and Skipper SNN with $p\%$ aggressive time-skipping respectively in (a), (c) & (e). In (b), (d) & (f) $x'-y'-z', (p_1\%-p_2\%-p\%)$ represents x',y' & z' nodes running on Skipper SNN with $p_1\%$, Skipper SNN with $p_2\%$ and Skipper SNN with $p\%$ aggressive time-skipping respectively.

TABLE II: Test accuracy for a 10/2 federated 4B-2C-2S set up for IID and Non-IID data.

Node Type	IID Data				Non- IID Data			
	FedAvg	wFedAvg [0.6, 0.4]	wFedAvg [0.75, 0.25]	wFedAvg [0.8, 0.2]	FedAvg	wFedAvg [0.6, 0.4]	wFedAvg [0.75, 0.25]	wFedAvg [0.8, 0.2]
4B4C2S 20%	86.34	86.21	86.42	86.11	81.45	83.72	80.53	81.9
4B4C2S 40%	84.78	85.83	84.94	84.85	82.87	82.42	81.98	79.24
4B4C2S 60%	85.09	83.71	86.13	86.36	80.85	81.87	80.31	83.10

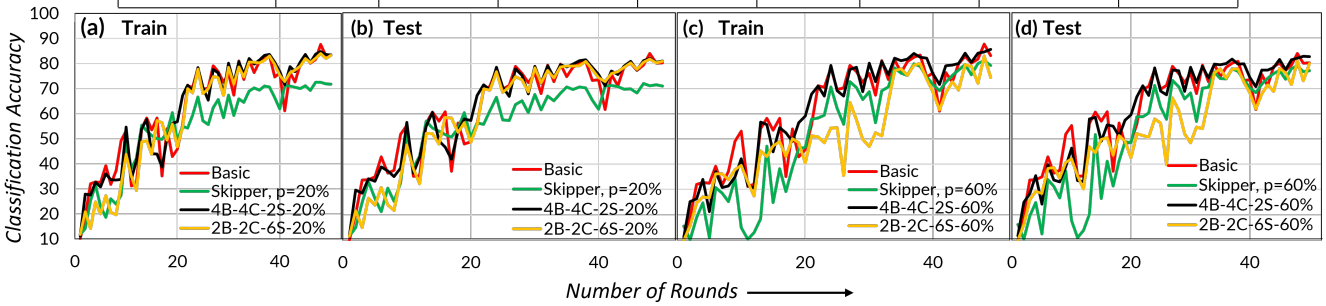

Fig. 5: Convergence plot for 10/2 *federated heterogeneous-clients* set-up with VGG9 on CIFAR10 *Non-IID* data. p = time-skipping fraction, $xB-yC-zS-w\%$ represents x,y & z nodes running on Basic, Checkpointed and Skipper SNN with $w\%$ time-skipping respectively.

of smart-sensors/processors deployed in a remote weather monitoring system and learning natively from local weather conditions would fit this category. In these experiments, some of the nodes have conservative time-skipping (p is small), whereas in others, time-skipping is more aggressive (p is large). Figures 4(b) and (d) show the test accuracy of this setup for VGG9 and LeNet, respectively. Again, the final test accuracy trends are similar to those observed in section VI-A – for conservative time-skipping (low p) or for smaller proportion of *memory-deficient* nodes, the heterogeneous variants outperform the homogeneous *skipper* counterparts, and also that their accuracies are competitive to homogeneous basic unrolled setup.

C. Heterogeneous FL with non-IID Data

In many practical federated learning scenarios, there is also heterogeneity in data distribution as dataset may not be uniformly distributed in all the clients. So, we experiment with non independent and identically distributed (non IID) data where the distribution of classes among clients is varied. We synthetically generate non-IID data using Dirichlet distribution with parameter $\alpha = 0.5$ to include skewness in class distribution among clients similar to the prior federated learning work [10], [25]. The memory configuration of the clients is the same as that in Section VI-A, i.e., we consider a mix of *memory-rich*, *memory-limited* and *memory-deficient* nodes, and compare them with homogeneous counterparts.

Figure 5 shows the training and validation convergence plots for 50 rounds of training. We observe that the convergence curves are smoother in the case of IID data compared to non-IID (refer Figure 3 and Figure 5). Nevertheless, each of the heterogeneous federated SNN learning setups (denoted as xB-yC-zS-$w\%$ where x is the number of basic (B) or fully unrolled SNNs, y is the number of checkpointed (C) SNNs and z is the number of *skipper* (S) SNNs with $p = w\%$) is able to converge well on the more challenging non-IID data. When the number of *skipper* nodes is small, the heterogeneous setup has a smoother curve and therefore, a better resilience to the high variance in the data, compared to the homogeneous setup (refer 4B-4C-2S-20% vs Basic and 4B-4C-2S-60% vs Basic). The same is true for

the 2B-2C-6S-20% setup as well (Figure 5 (a) and (b)), but not for the 2B-2C-6S-60% setup (Figure 5 (c) and (d)) We infer that the heterogeneous framework is more robust in handling the variability in non-IID data compared to the homogenous baseline, as long as it has either few *memory-deficient* nodes or many *memory-deficient* nodes and a conservative time-skipping percentage (low p). We further note that the homogeneous setup with all *memory-deficient* nodes has a slower convergence and higher sensitivity to non-IID data compared to its heterogeneous counterparts (*skipper*, $p = 20\%$). This further corroborates our hypothesis that weak nodes can be supported by stronger nodes and enabled to participate in global model updates and achieve competitive accuracy compared to when they are running in a homogeneous environment.

Figures 4(e) and (f) report the final test accuracy of VGG9 SNN in different mixed-memory and in different *memory-deficient* heterogeneous FL scenarios, respectively, and compare it with that of the homogeneous settings for *non-IID* data. We see that the heterogeneous variants closely match the basic and checkpointed SNN accuracy and the heterogeneous framework outperforms the homogeneous *memory-deficient* counterpart when p is low or the number of *memory-deficient* nodes is low. A similar trend is also observed for the *memory-deficient* heterogeneous FL scenarios in Figure 4(f). However, this trend tends to reverse for aggressive time-skipping (large p) or for larger proportion of the *memory-deficient* nodes. Finally, note that, the memory consumption of the SNNs will not be affected by the changes to the data distribution (see Table I for these measurements on VGG9 with CIFAR10).

D. Training-type based weighted averaging aggregation technique

In this set of experiments we experimented with different weight-proportion ($\omega^{(n)}$) for weighted federated averaging (*wFedAvg*) as described in section IV and the final test accuracies for heterogeneous FL with 4B-4C-2S mixed memory clients set-up with $20\%, 40\%$ and 60% skipper model are reported in Table II for both IID and Non-IID data distribution at edge devices. For a 10/2 federated set-up *FedAvg* is equivalent to setting equal proportion for two participating clients, which means using [0.5, 0.5] as weight-proportion list. We then experimented with varing weight-proportion like [0.6, 0.4], [0.75, 0.25] etc. as mentioned in Table II to investigate the impact of ($\omega^{(n)}$) for *wFedAvg*. No perceptible difference nor any trend is observed if we vary the proportion across different client based on their training-type. Since data is distributed in heterogeneous nodes, they have contributed well enough to give reasonable performance in this heterogeneous federated set up. Hence, weight-proportion is not very critical to the global model aggregation. So we have performed all other experiments with the prior approach (*FedAvg*).

VII. Conclusion

We explored a federated learning framework with memory efficient SNN and observed that the total memory requirements for training can be reduced by $\sim 4\times$ compared to training with unrolled SNN. We further showed how, even low memory capacity devices, which would otherwise have been incapable of learning, can now contribute to SNN training in a federated capacity. Then, we investigated how this memory efficient SNN can be leveraged for the heterogeneity of edge clients in a federated learning framework, with *memory-rich*, *memory-limited*, and *memory-deficient* type of clients, for both the IID and Non-IID data distributions with two different aggregation approaches. We observed competitive performance of heterogeneous-clients based federated learning framework as its homogeneous counterpart. For the non-IID data, the *memory-deficient* nodes are supported by other *memory rich* or stronger nodes that help to improve the test accuracy up to 10%.

References

[1] S.-C. Liu and T. Delbruck, "Neuromorphic sensory systems," *Current Opinion in Neurobiology*, vol. 20, no. 3, pp. 288–295, 2010.

[2] A. Vanarse, A. Osseiran, and A. Rassau, "A review of current neuromorphic approaches for vision, auditory, and olfactory sensors," *Frontiers in Neuroscience*, vol. 10, 2016.

[3] M. Bouvier, A. Valentian, T. Mesquida, F. Rummens, M. Reyboz, E. Vianello, and E. Beigne, "Spiking neural networks hardware implementations and challenges: A survey," *ACM JETC*, 2019.

[4] M. Davies, N. Srinivasa, T. H. Lin, G. Chinya, P. Joshi, A. Lines, A. Wild, and H. Wang, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE MICRO*, vol. PP, no. 99, pp. 1–1, 2018.

[5] M. Davies, A. Wild, G. Orchard, Y. Sandamirskaya, G. A. F. Guerra, P. Joshi, P. Plank, and S. R. Risbud, "Advancing neuromorphic computing with loihi: A survey of results and outlook," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 911–934, 2021.

[6] Brainchip Holding Ltd., "Akida neuromorphic system-on-chip," [Accessed: Mar-30-2023].

[7] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th AISTATS*, ser. Proceedings of Machine Learning Research. PMLR, 2017, pp. 1273–1282.

[8] L. Li, Y. Fan, M. Tse, and K.-Y. Lin, "A review of applications in federated learning," *Computers & Industrial Engineering*, 2020.

[9] "Amazon Alexa," https://developer.amazon.com/en-US/alexa, Accessed: Mar-30-2023.

[10] Y. Venkatesha, Y. Kim, L. Tassiulas, and P. Panda, "Federated learning with SNNs," *IEEE Transactions on Signal Processing*, 2021.

[11] P. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[12] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in snns: Bringing the power of gradient-based optimization to SNNs," *IEEE Signal Process. Mag.*, 2019.

[13] S. Singh, A. Sarma, S. Lu, A. Sengupta, M. T. Kandemir, E. Neftci, V. Narayanan, and C. R. Das, "Skipper: Enabling efficient snn training through activation-checkpointing and time-skipping," in *IEEE MICRO*, 2022, pp. 565–581.

[14] N. Skatchkovsky, H. Jang, and O. Simeone, "Federated neuromorphic learning of snns for low-power edge intelligence," in *ICASSP*, 2020.

[15] H. Jang, O. Simeone, B. Gardner, and A. Gruning, "An introduction to probabilistic spiking neural networks: Probabilistic models, learning rules, and applications," *IEEE Signal Processing Magazine*, 2019.

[16] H. Yang, K.-Y. Lam, L. Xiao, Z. Xiong, H. Hu, D. Niyato, and H. Vincent Poor, "Lead federated neuromorphic learning for wireless edge artificial intelligence," *Nature communications*, 2022.

[17] X. Cheng, T. Zhang, S. Jia, and B. Xu, "Finite meta-dynamic neurons in spiking neural networks for spatio-temporal learning," *arXiv preprint arXiv:2010.03140*, 2020.

[18] S. Subbulakshmi Radhakrishnan, A. Sebastian, A. Oberoi, S. Das, and S. Das, "A biomimetic neural encoder for snn," *Nature communications*, 2021.

[19] Y. Kim and P. Panda, "Revisiting batch normalization for training low-latency deep SNNs from scratch," *Frontiers in Neuroscience*, 2021.

[20] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," *IEEE ICC*, pp. 1–7, 2019.

[21] E. Diao, J. Ding, and V. Tarokh, "HeteroFL: Computation and communication efficient federated learning for heterogeneous clients," in *ICLR*, 2021.

[22] S. Singh, A. Sarma, N. Jao, A. Pattnaik, S. Lu, K. Yang, A. Sengupta, V. Narayanan, and C. Das, "Nebula: A neuromorphic spin-based ultra-low power architecture for snns and anns," in *ACM/IEEE ISCA*, 2020.

[23] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski, *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

[24] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine learning and systems*, vol. 2, pp. 429–450, 2020.

[25] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, "Federated learning with matched averaging," 2020. [Online]. Available: https://arxiv.org/abs/2002.06440

[26] S. Singh, A. Sarma, S. Lu, A. Sengupta, V. Narayanan, and C. R. Das, "Gesture-snn: Co-optimizing accuracy, latency and energy of snns for neuromorphic vision sensors," in *ACM/IEEE ISLPED*, 2021.