HFedSNN: Efficient Hierarchical Federated Learning using Spiking Neural Networks

Ons Aouedi, Kandaraj Piamrat, Mario Südholt

HFedSNN: Efficient Hierarchical Federated Learning using Spiking Neural Networks

Ons Aouedi, Kandaraj Piamrat and Mario Südholt
firstname.lastname@ls2n.fr
IMT Atlantique, Nantes University, École Centrale Nantes, CNRS, INRIA, LS2N, UMR 6004
F-44000 Nantes, France

ABSTRACT

Federated Learning (FL) has emerged in edge computing to address privacy concerns in mobile networks. It allows the mobile devices to collaboratively train a model while keeping training data where they were generated. However, in practice, it suffers from several issues such as (i) robustness, due to a single point of failure, (ii) latency, as it requires a significant amount of communication resources, and (iii) convergence, due to system and statistical heterogeneity. To cope with these issues, Hierarchical FL (HFL) has been proposed as a promising alternative. HFL adds the edge servers as an intermediate layer for sub-model aggregation, several iterations will be performed before the global aggregation at the cloud server takes place, thus making the overall process more efficient, especially with non-independent and identically distributed (non-IID) data. Moreover, using traditional Artificial Neural Networks (ANNs) with HFL consumes a significant amount of energy, further hindering the application of decentralized FL on energy-constrained mobile devices.

Therefore, this paper presents HFedSNN: an energy-efficient and fast-convergence model by incorporating Spike Neural Networks (SNNs) within HFL. SNN is a generation of neural networks, which promises tremendous energy and computation efficiency improvements. Taking advantage of HFL and SNN, numerical results demonstrate that HFedSNN outperforms FL with SNN (FedSNN) in terms of accuracy and communication overhead by 4.48% and 26×, respectively. Furthermore, HFedSNN significantly reduces energy consumption by 4.3× compared to FL with ANN (FedANN).

CCS CONCEPTS

• **Computer science** → **Spiking Neural Networks**; • **Distributed System** → *Hierarchical Federated Learning*; • **Networks** → Mobile networks.

KEYWORDS

Federated learning, hierarchical federated learning, spiking neural networks, energy saving.

ACM Reference Format:
Ons Aouedi, Kandaraj Piamrat and Mario Südholt. 2023. HFedSNN: Efficient Hierarchical Federated Learning using Spiking Neural Networks. In . ACM, New York, NY, USA, 8 pages. https://doi.org/XXXXXXX.XXXXXXX

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

To address the limitations associated with centralized learning and local training, Federated Learning (FL) has emerged as a compelling alternative learning paradigm, primarily because of its ability to preserve privacy. FL empowers edge devices to learn collaboratively, ensuring that sensitive private data remain secure and are not shared with external entities. The growing demand for mobile networks presents a challenge for cloud FL in terms of privacy, security, latency, and bandwidth. In particular, although it implicitly offers a certain degree of privacy because sensitive data are not exposed, some limitations exist, such as the problem of heterogeneous environments, including hardware and statistical heterogeneity. For instance, the diversity of data generated in mobile networks leads to non-independent and identically distributed (non-IID) data, meaning that the data across each client vary in size and distribution. This can potentially cause divergence in the FL model and in turn, decrease its performance. This problem is amplified further as the number of clients increases. In contrast, FL assumes that the server residing in the cloud is an FL server for model aggregation. However, communication between mobile devices and cloud servers may be frequently unavailable, expensive, and slow. This has motivated researchers to take advantage of both edge and cloud servers and propose a new paradigm called Hierarchical Federated Learning (HFL) [7].

Although the performance of HFL reduces the impact of non-IID data on the model performance, its training process occurs on limited computing and low-energy devices. This makes the participation of such constrained devices in the FL process almost impossible. To solve this issue spiking neural networks (SNNs) have recently become very popular as an energy-efficient alternative for artificial intelligence tasks. SNNs are also known as event-driven spiking networks and form a new generation of neural networks [8]. The intrinsic operational characteristics of SNNs make them power efficient. Unlike Artificial Neural Networks (ANNs), SNNs replace the multiplicative operations of inputs and weights with simpler addition operations. This fundamental difference leads to a reduction in the power consumption of SNN-based models, offering a more energy-efficient alternative to traditional ANNs [13]. Therefore, SNNs are suitable for training many machine learning tasks in mobile networks with HFL.

In this paper, we propose a novel HFL framework that incorporates SNNs, called `HFedSNN`. This approach leverages a client-edge-cloud architecture wherein each mobile device (client) independently trains its SNN model on locally private data. To the best of our knowledge, this is one of the first studies that explore SNN-based models within an HFL context.

1.1 Contributions

In this paper, we present the following contributions:

- We propose an HFL framework to train energy-efficient SNNs on mobile devices in a privacy-preserving way. The SNNs are suitable and practical for HFL in green cloud/edge systems owing to their advantages of low power consumption and event-driven information processing.
- We conduct extensive experiments to show the energy efficiency and classification performance of HFedSNN with IID and non-IID data on a real dataset by comparing it with baseline schemes. The numerical results demonstrate the advantages of the proposed model, particularly with non-IID data, energy efficiency, and communication cost.
- We analyze the effect of several important hyper-parameters for HFL and SNNs on classification performance.

The rest of the paper is organized as follows. Section 2 summarizes the work related to this paper. Section 3 presents essential background information and our HFedSNN approach. Experimental settings and results are presented in Section 4. Finally, the conclusion is given in Section 5.

2 RELATED WORK

SNNs have recently emerged as a new generation of low-energy deep neural networks thanks to their binary and event-driven processing nature. The potential of SNNs has been explored in various domains including human activity recognition [6] and object detection [4]. For example, Li et al. [6] proposed SNNs for human activity recognition for the first time. The experimental results demonstrated that SNNs not only exceeded ANNs in terms of accuracy but also reduced energy consumption by up to 94%. Yan et al. [13] used SNNs to classify Electrocardiogram (ECG) beats in energy-constrained wearable devices. However, despite the performance of the proposed model, the authors used ANN-SNN conversion where the convolutional neural network (CNN) model was first trained, and the resulting weights were transferred to SNNs with the same network structure. However, this conversion process leads to an increase in the overall energy consumption of the model.

These approaches typically require a central entity to process data collected from all users. Therefore, operators face difficulties in collecting large amounts of data, especially if end users are not willing to share their sensitive data. Consequently, FL has been used as an alternative solution to privacy concerns. Several research efforts have been devoted to the use of FL with SNN. In this context, Skatchkovsky et al. [10] proposed an FL-based learning model for networks on an SNN device called FL-SNN. The experimental results show a flexible trade-off between communication cost and accuracy. However, only two clients were used during the training. Similarly, Xie et al. [12] presented an efficient FL with an SNN for traffic sign recognition in networked vehicles. The simulation results showed that federated SNN outperformed traditional federated convolutional neural networks in terms of accuracy and energy efficiency. In contrast to previous studies, which were limited to extracting computation and energy efficiency with the spiking model, the Venkatesha et al. [11] studied the scalability and robustness of SNN with FL. The results showed that federated SNN outperformed federated ANN when the data were distributed across a large number of clients and provided up to 4.3× energy efficiency. However, using the cloud as an FL server cannot satisfy delay-sensitive applications, and the model performance is heavily influenced by non-IID data distributions [7]. To avoid total reliance on cloud servers, the authors of Yang et al. [14] proposed a decentralized Federated SNN model without using a fixed central coordinator, called LFNL. The results also showed a significant reduction in energy consumption compared with the federated ANN. Although the proposed model eliminates the need for a central server, it is still expensive in terms of communication overhead.

Discussion: While the integration of SNNs with FL can significantly reduce energy consumption, it is important to note that both peer-to-peer FL and classical FL are limited by the speed of convergence. This leads to high consumption of computational resources by end devices along with communication overhead by the corresponding approaches. Considering these factors, our focus is on combining SNNs with HFL. This approach not only offers the advantages of efficient distributed learning with non-IID data but also ensures a reduction in communication costs and energy consumption, thus making it an efficient and promising solution.

3 THE HFedSNN PROPOSAL

As background, we first present the two main concepts that have been used in our proposal, HFL, and SNN, followed by the methodology and the architectural design of our model.

3.1 Concepts

3.1.1 Hierarchical Federated Learning (HFL). In contrast to traditional FL, HFL employs several aggregations of local models taking place at the edge. This is followed by sending the edge aggregated sub-models to the cloud for global aggregation. For the aggregation, the FederatedAveraging (FedAvg) algorithm [9] has been used.

$$w_{t+1} = \sum_{k=1}^{P} \frac{D_k}{D} w_{t+1}^k \tag{1}$$

where the K clients are indexed by k, w is the model parameter at communication round $t+1$, D is the total amount of the data in all the participants P (P<K), and D_k is the training data available to client k.

Using the edge devices as intermediate servers not only helps to improve the model performance but also enables efficient communication, enhances the robustness and flexibility of large-scale networks, as well as reduces the latency to meet delay-sensitive application requirements.

3.1.2 Spiking Neural Networks (SNNs). SNNs are a biologically inspired neural network, in which the neurons process spike signals over time, rather than real numbers (Figure 1). The sparsity of the synaptic spiking inputs and its event-driven nature offer significant energy reduction compared to conventional artificial neural networks (ANNs). In particular, the energy consumed by the SNN-based model during learning and inference is essentially proportional to the number of spikes processed and communicated by the neurons. The spikes are emitted when the membrane potential exceeds the pre-defined threshold. For example, as illustrated in Figure 1, with the Leaky Integrate-and-Fire (LIF) neural model

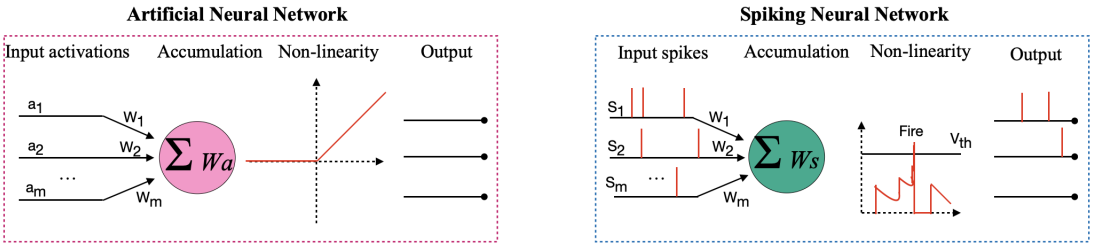


Figure 1: The structure of ANN and SNN. The ANN takes input and rectifies it if is less than 0 and passes it otherwise. The SNN takes spikes as input and fires as spikes only if the membrane potential exceeds the threshold V_{th}.

(used in our work) the neuron accumulates the input spikes (s_1, s_2, ..., s_m), called membrane potential and generates a spike output whenever $\sum Ws$ exceeds the firing threshold V_{th}. The accumulated spike value will continue to increase each time the neuron fires and gradually decay toward a resting value when the neuron is not firing due to a leak factor. After the neuron fires, the membrane potential is lowered by the amount of the threshold. This process is repeated for T timesteps.

For a single post-synaptic neuron i, the potential membrane can be represented as follows:

$$u_i^t = \lambda u_i^{t-1} + \sum w_{ij} o_j^t \tag{2}$$

where j represents the pre-synaptic neuron, λ is the leak factor, o_j the binary spike activation, and w_{ij} the weight of the connection between the neurons j and i.

Unlike conventional ANNs, the information in SNNs (i.e., the training data) needs to be encoded as a set of spike sequences with time steps using either a *rate-based* method, *temporal coding*, or *direct coding* (for more details of encoding techniques see [2]). Furthermore, since SNNs involve non-differentiable functions, the widely-used gradient-based backpropagation cannot be used directly to train it. To address this issue, several training techniques have been proposed, that can broadly be categorized into three types: (i) Spike-Timing Dependent Plasticity (STDP), (ii) Conversion (ANN-SNN), and (iii) Surrogate Gradient Descent (SGD). With STDP, if a pre-synaptic neuron fires just before a post-synaptic neuron, the weight between those two neurons is increased. Alternatively, if a pre-synaptic neuron fires just after a post-synaptic neuron, the weight between those two neurons is decreased. With ANN-SNN conversion, the researchers used normalization methods to transfer ReLU activation to integrate and fire spiking activity. Finally, SGD is a continuous and differentiable approximation of the non-differentiable spike function. This training technique enables SNNs to be trained from scratch with lower latency and reasonable classification accuracy.

3.2 Methodology & Architecture

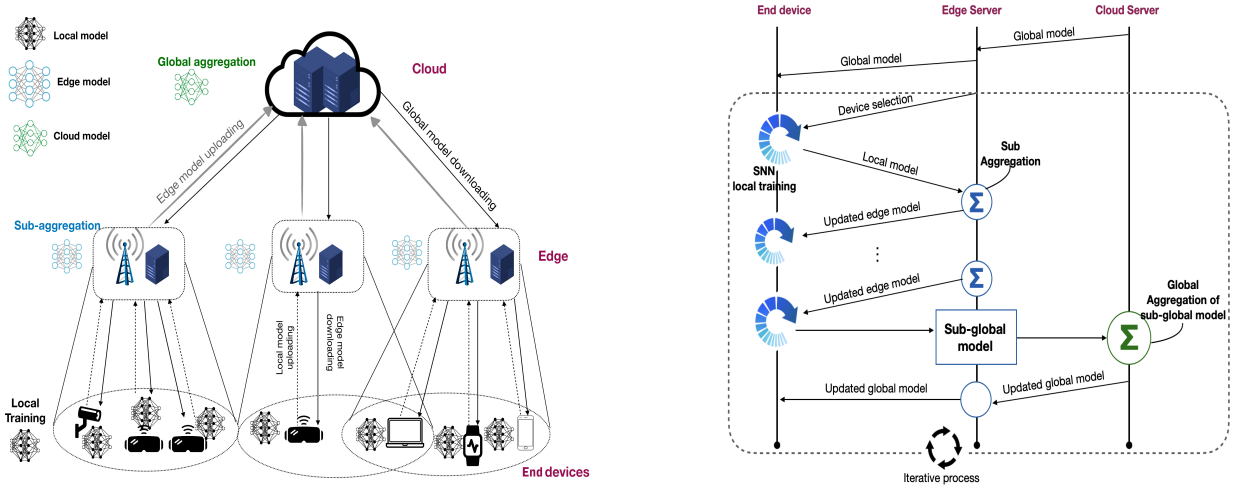
To achieve better learning efficiency of SNNs in this study, we propose HFedSNN, which combines the strengths of HFL and SNN-based models. Our HFedSNN is designed to optimize the learning process, particularly with non-IID data, ensuring low communication overhead and energy consumption. Figure 2 and Algorithm 1 present the architecture and main procedure of HFedSNN. Our framework consists of three layers and its learning process includes the following key steps.

(1) *Distributed local training and updates*: Once the subset of the clients (e.g., mobile devices) that participate in the learning process is selected, the cloud server sends an initial SNNs model (VGG9 in SNNs version) to them to trigger the distributed training (*Global model download*). Then, after some local iterations, each client sends its local model updates to the corresponding edge server for sub-global model aggregation (*Local model upload*).
(2) *Sub-global model aggregation and upload*: Upon receiving all the updates from the participants, the edge servers perform the sub-global model aggregation and transfer it back (*Edge model downloading*) to their assigned devices to update their local models accordingly. Then, after a specific number of sub-global rounds, the edge servers send their sub-global models (*Edge model uploading*) to the cloud server.
(3) *Global model aggregation*: After receiving the sub-global models, a combined global model is created by averaging the parameters of the edge models. Finally, the global model parameters are transmitted along the hierarchy downwards to the mobile devices.

This process is iterated until the desired accuracy is achieved. In alignment with the methodology of [11], we encode the data prior to initiating the training process. This involves transforming pixel values into spike trains of certain timesteps using the Poisson rate coding. Similarly, for gradient-based training within our SNNs model, we utilize Batch Normalization Through Time (BNTT) [5].

4 EXPERIMENTS AND RESULTS

In this section, we first present our experiment settings, and then the corresponding results along with their analysis. We use Python as a programming language and PyTorch to construct our model. Some basic code was adopted from the studies by Venkatesha et al. [11]. It is important to highlight that our objective is to primarily use HFedSNN for a comparison study with the FedSNN model, as detailed in [11]. We do not aim to achieve optimal accuracy with this model in this paper, but rather to understand its performance in relation to FedSNN.



(a) HFL framework within mobile networks **(b) Communication process**

Figure 2: The network architecture and communication process of our HFedSNN model.

4.1 Simulation Settings

Benchmarks: We compare our proposed model with three baseline approaches for performance evaluation in terms of accuracy, energy consumption, and communication cost. The first one is *FedSNN* proposed by Venkatesha et al. [11], which is a cloud-based FL using the SNNs model. We also choose an FL-based ANN model (*FedANN*) and an HFL-based ANN (*HFedANN*) as the baselines, which train the VGG9 model in the ANNs version.

Models and Datasets: The experiments have been conducted over a model (i.e., VGG9 in SNNs version) and the CIFAR-10 dataset. VGG9 consists of 7 convolutional layers and two fully connected layers. CIFAR-10 is a real dataset including 50,000 images for training and 10,000 for testing and has ten different types of objects. We choose the same model and dataset to compare with the baseline [11].

Simulation Parameters: In alignment with the methodology of [11], in all simulations, we use BNTT as a surrogate gradient-based backpropagation approach and we encode the pixel values into spike trains of length T using rate coding. Furthermore, we use 8 rounds as sub-global rounds and 5 rounds as global rounds (R_g) so in total, we run 40 communication rounds (on two levels). We select these values for a fair comparison with `FedSNN` [11] because it used 40 communication rounds; we also use similar values for the total number of clients K, participants P, timesteps T, and local epoch on each client. To further evaluate the performance of our model, we conduct experiments on several numbers of clients $K \in \{100, 150, 200\}$, participants $P \in \{10, 15, 20\}$, and timesteps $T \in \{20, 25, 30\}$ in each communication round to study their impact on the performance. Also, to analyze the impact of non-IID data on the performance of our model, we vary the α parameter of the Dirichlet distribution. Table 1 summarizes the `HFedSNN` parameters and their selected settings in our simulations.

Table 1: Selected parameters

Spiking Neural Network	
Library	PyTorch
model	VGG9
Timesteps	25
Threshold	1
Hierarchical Federated Learning	
FL server	1
Edge devices	3
Total clients	100
Participants	10%
Edge rounds (clients-edge devices)	8
Global round (edge-devices-FL server)	5
Local epoch (client)	5
α	0.5

4.2 Classification performance

Figure 3 shows the classification accuracy of our model against `FedSNN` [11] and `FedANN` for IID and non-IID data. Our model demonstrated superior accuracy compared with `FedSNN`, particularly in the non-IID data scenario. This enhanced performance was achieved with significantly fewer global communication rounds (Figure 4), requiring only five global rounds, as opposed to the 40 global rounds necessary for `FedSNN`, that is, an 8-fold improvement. This is mainly due to the use of edge devices for sub-model aggregation with SNNs. The intermediate layer not only accelerates the convergence of the global model but also mitigates the impact of non-IID data with only local communication costs.

4.3 Impact of data distribution

In the context of mobile networks, data are often distributed unevenly and differently among devices for various reasons such as

Algorithm 1: Learning procedure of HFedSNN: R_g is the total number of global rounds, r is the fraction of selected participants in each round. Edge servers C are indexed by c and K_c is the client associated with the edge server c. On the client, D_k is the dataset available to client k, E is the number of local epochs, and η is the learning rate. T is the total timestep, and V_{th} represents the firing threshold. B is the local batch size.

/* Edge and Cloud servers side */
procedure HFLSNN
  Initialise w_0 globally
  for each global round $g \in [1, R_g]$ **do**
    for each edge server $c \in C$ **do *in parallel***
      Initialise $w_{c,g,0} \longleftarrow w_g$
      for each local round $t \in [1, R_c]$ **do**
        $w_{c,g,t+1} \longleftarrow \text{FedSNN}(w_{c,g,t}, K_c)$
      end for
    end for
    $w_{g+1} \longleftarrow FedAvg(w_{c,g,R_c})$
  end for
end procedure

procedure FedSNN(w_t, K_c)
  $P_c = r \times |K_c|$
  for each client $k \in P_c$ **do *in parallel***
    $w_{t+1}^k \longleftarrow \text{ClientUpdate}(k, w_t, T, V_{th})$
  end for
  $w_{t+1} \longleftarrow FedAvg(w_{t+1}^k)$
end procedure

/* Client side */
procedure ClientUpdate(k, w, T, V_{th})
  $\beta \longleftarrow$ (Split D_k into batch of size B)
  for each batch $b \in \beta$ **do**
    for each epoch $t \in [1, E]$ **do**
      Update w using SNN(b, w, V_{th}, η)
    end for
  end for
  return $\Delta w = w - w_0$
end procedure
procedure SNN(b, w, V_{th}, η)
  Initialise neuron potentials $V_i = 0$ for all neurons i
  for each timestep $t \in [1, T]$ **do**
    Compute input current $I = w^T \cdot b$
    Update neuron potentials V_i based on I_i
    If $V_i > V_{th}$ for any neuron i, emit spike and reset V_i
    Update weights w based on emitted spikes and learning rule, with learning rate η
  end for
end procedure

user behavior and device characteristics. In this subsection, we examine the impact of non-IID data on the performance of our model. With IID settings, each client holds $\frac{D}{K}$ training samples. The model was trained with 100 clients and 10 participants, with

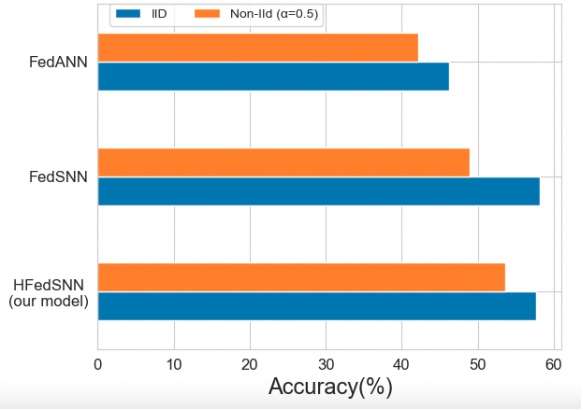


Figure 3: The classification performance (accuracy) of our model (with 5 global rounds) against FedSNN [11] and FedANN models with 40 global rounds, on IID and non-IID data with 100 clients and 10 clients participating.

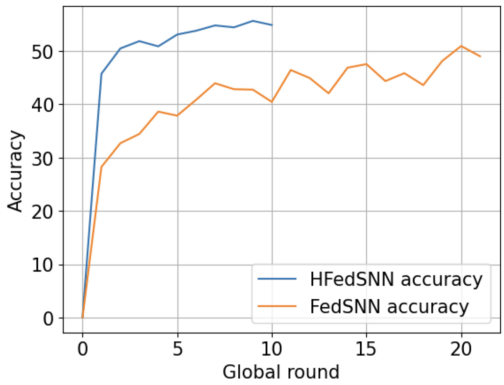


Figure 4: Model convergence with non-IID data settings

each client holding 500 samples. For non-IID settings, we study the parameter α of the Dirichlet distribution [15], where $\alpha \to \infty$, all clients have an IID data distribution, and with $\alpha \to 0$ each client holds samples from only one randomly chosen class. As shown in Figure 5, non-IID data decreased the performance of both HFedSNN and FedSNN. However, our model exhibited superior performance when dealing with non-IID data (a lower level indicates a higher degree of non-IID data). In particular, when $\alpha = 2$ the accuracy of our model is 0.6% better than that of FedSNN, whereas $\alpha = 0.5$ the accuracy of our model is 4.48% better than that of FedSNN. This was attributed to the intermediate edge layer used in the model.

4.4 Energy consumption analysis

Energy consumption can be estimated based on the number of floating point operations (FLOPs) of ANNs or SNNs, which is approximately equivalent to the number of multiply-and-accumulate (MAC) operations. To evaluate the energy consumption of HFedSNN, we used the metric described in [3] and the formulas proposed

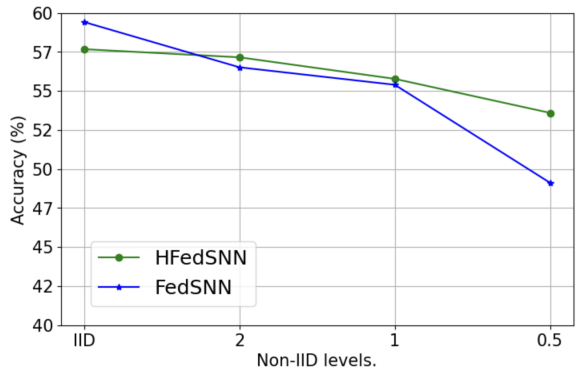


Figure 5: Classification accuracy under different non-IID levels.

in [12]:

$$E_{ANN} = FLOPs(l) \times (E_{Mult} + E_{Add}) \tag{3}$$

$$E_{SNN} = FLOPs(l) \times R_s(l) \times T \times E_{AC} \tag{4}$$

where E_{MulT} and E_{Add} represent the estimated energy of MAC operations. E_{AC} represents the estimated energy of accumulated operation (AC). $R_s(l)$ is the spike rate of the $l-th$ layer. Given the number of spikes n_l^s sent by the neurons in $l-th$ layer within timesteps T and the number of neurons N_l in the $l-th$ layer, $R_s(l)$ can be defined as:

$$R_s(l) = \frac{n_l^s}{N_l} \tag{5}$$

Referring to equations 3 and 4, we obtain the total energy estimate for the VGG9-based SNNs model as approximately 53.24μJ. In contrast, the VGG9-based ANN model required approximately 227.99μJ, making the SNNs model **4.3**× more energy-efficient. The main reason for this significant improvement is the binary propagation process in SNNs, which performs accumulation (AC) operations, thereby reducing energy consumption. Optimizing energy consumption can lead to energy-efficient mobile networks that ensure better network availability and performance as well as contribute to more sustainable digital infrastructures.

4.5 Communication overhead

Because communication overhead is a crucial criterion in mobile networks, we compared this factor in our model against the HFedANN, FedSNN, and FedANN models. In particular, we used the formulas proposed by Aouedi et al. [1] where the communication cost of these models is calculated as follows:

$$C = P \times R_g \times (2 \times size(H)) \tag{6}$$

where P is the number of participants in each communication round, R_g is the total number of global rounds, and $size(H)$ is the size of the model located on the client and exchanged between the edge servers and cloud server in each communication round. We measured the size of the model of VGG9 in the SNN/ANN version as the size of the saved PyTorch model state_dict file.

Figure 6 presents the communication efficiency of HFedSNN compared with the existing models presented in Figure 6. It shows the total communication cost of the entire training process for each model using the required communication rounds. The results demonstrate that HFedSNN significantly reduces communication resources by **99**×, **26**×, and **3.7**× compared to FedANN, FedSNN, and HFedANN, respectively. This implies that integrating SNNs into HFL is a cost-effective strategy and, in turn, can improve the quality of services (QoS) provided to users and provide low-latency services.

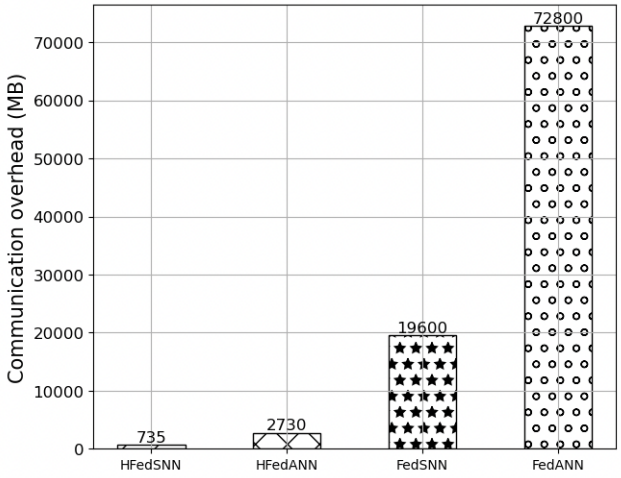


Figure 6: Comparison in terms of communication overhead (Lower implies better).

4.6 Impact of HFL parameters

In this subsection, we study the impact of some HFL parameters on the overall accuracy of the model. In the following, we consider the number of clients and edge servers.

Number of total/participant clients. In this subsection, we investigate the effect of varying the total number of clients, denoted by K on the classification performance of HFedSNN. While maintaining a constant fraction of participating clients, we adjust K, which alters the total number of participating clients. This analysis aims to understand how the model's performance scales with an increase in both the total and participating clients.

As shown in Table 2, it is clear that an increase in the total number of clients adversely affects the performance of both HFedSNN and FedSNN [11] in both the IID and non-IID data settings. Even though the performance of FedSNN is better than ours in IID settings, our model provides superior results in non-IID settings, even when the number of clients is large, which is the objective of using HFL.

Number of Edge Servers. Edge servers are important in HFL and can lead to significant improvements in system performance, scalability, privacy, and fault tolerance. As shown in Figure 7, increasing the number of edge servers in the intermediate layer can enhance the classification performance of our model in both the non-IID and IID data settings. This is because by increasing the number of edge servers, each server will be responsible for a smaller group of

Table 2: Classification accuracy with different numbers of clients (K) and participants (P).

Clients (K/P)	IID		Non-IID	
	FedSNN [11]	HFedSNN	FedSNN [11]	HFedSNN
100/10	59.42	57.68	49.12	53.60
150/15	59.86	53.29	50.51	53.61
200/20	59.26	51.70	50.15	51.43

clients, thus reducing the impact of non-IID data and potentially improving the classification accuracy of the global model. However, it should be noted that a large number of edge servers increases the complexity of the system and leads to higher infrastructure costs. Therefore, determining the trade-off between classification accuracy and complexity of systems is an important research direction.

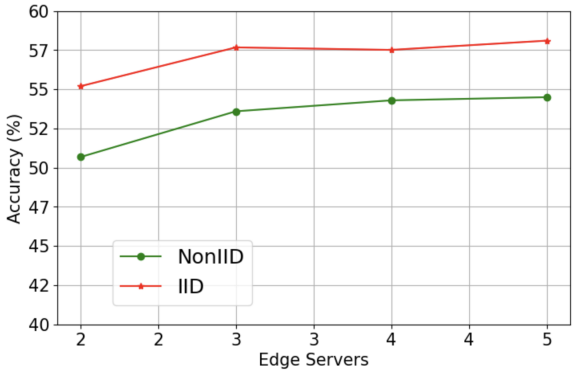


Figure 7: Classification accuracy of HFedSNN with different numbers of edge servers

4.7 Impact of SNNs hyperparameters

In this subsection, we study the impact of some SNNs hyperparameters on the overall accuracy of the model. We consider the timestep and leakage rate parameters.

Timestep. As illustrated in Figure 8, there is a direct relationship between the increase in the timestep and the improvement in accuracy. This is because a larger number of spikes provides a more comprehensive representation of the data. However, increasing the timestep also leads to a proportional increase in the time required for the SNNs to converge, thereby affecting the latency of the system.

Leakage rate. The leakage factor denoted as $\lambda \in [0, 1]$ corresponds to the leakage of the membrane potential u. In particular, it represents the rate at which the neuron decays over time when it is not firing or receiving any input. Thus, the leaky factor can also determine how long a neuron can remember the past data. As shown in Figure 9, the leakage rate can significantly affect SNNs learning process and the performance of SNNs. For example, a large leakage rate implies that the membrane potential of the neurons decays quickly (Equation 2) and, in turn, reduces spike production unless

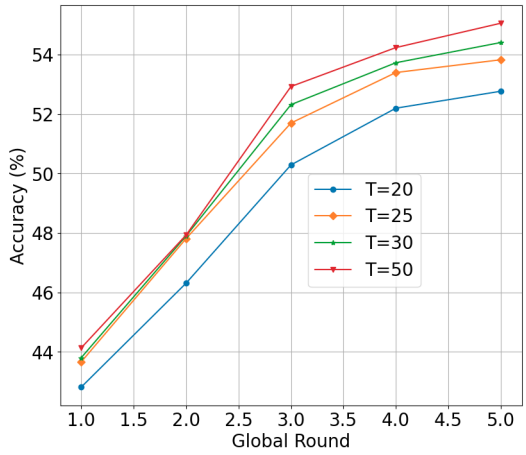


Figure 8: Performance comparison with different timesteps

it receives frequent input. Similarly, a small leakage rate implies that the neurons are too active; hence, the final model is prone to overfitting. Through experimental verification, we observed that $\lambda = 0.5$ was the optimal setting for HFedSNN.

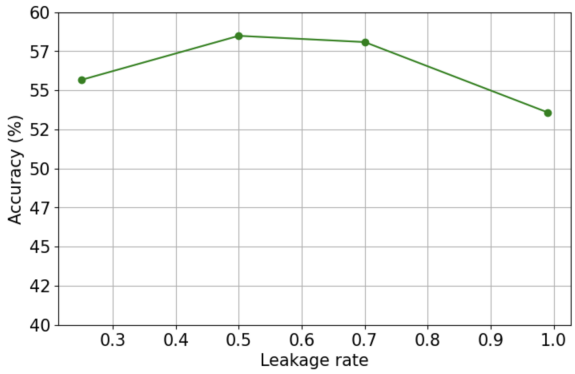


Figure 9: Classification performance with different leak rates

4.8 Experiments on the second dataset

We also tested our model against FedSNN using both non-IID and IID data settings on the MNIST dataset, which is an open and well-known dataset. The simulation results, shown in Figure 10, indicate the superior performance of our model over FedSNN under both scenarios. Furthermore, our model achieved an accuracy of over 75% and 99% under non-IID and IID conditions, respectively. These results demonstrate that integrating HFL and SNN can notably enhance the model's performance in terms of classification accuracy, convergence speed, communication cost, and most importantly, energy consumption. Furthermore, our model exhibits robust performance under both non-IID and IID settings, suggesting its scalability. Thus, the results demonstrate that the proposed model can be a viable solution for mobile networks with respect to their specific requirements and constraints.

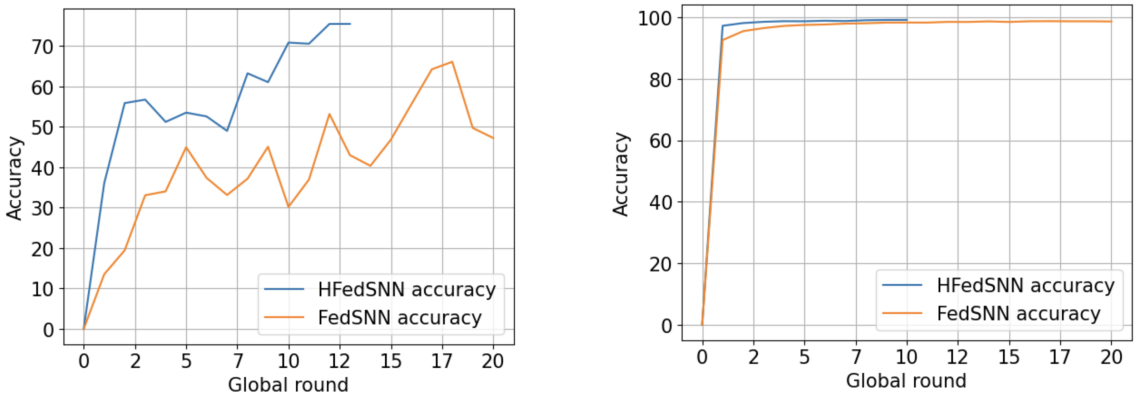


Figure 10: Classification accuracy (a) non-IID and (b) IID settings of data distribution with MNIST dataset

5 CONCLUSION

In this study, we have proposed and analyzed a hierarchical federated spiking neural network (HFedSNN) model, which uses SNNs for the local training of mobile devices. Unlike conventional Federated Learning (FL) approaches, our model integrates an edge layer for intermediate aggregation prior to global aggregation on the cloud server. We have evaluated the proposed classification capabilities of HFedSNN under both IID and non-IID data settings against several baseline approaches while varying different parameters. In addition, we have investigated the performance of our model by varying several parameters including the number of clients, time steps, and non-IID levels. Experimental results using real-world datasets have shown that the introduction of an intermediate edge layer can improve the performance of the final model under non-IID data conditions and reduce communication costs. Furthermore, the numerical results of the SNNs revealed that the proposed HFedSNN model achieved significant reductions in energy consumption.

In the future, we plan to incorporate clustering algorithms, such as Hierarchical Clustering into our HFedSNN model. This integration aims to further improve model performance, especially with non-IID data settings. By organizing data into clusters based on their similarities, we can potentially mitigate the challenges associated with non-IID data and obtain a more accurate and efficient model. Moreover, we plan to personalize global models to better work for individual clients.

REFERENCES

[1] Ons Aouedi, Kandaraj Piamrat, Guillaume Muller, and Kamal Singh. 2022. Federated semisupervised learning for attack detection in industrial Internet of Things. *IEEE Transactions on Industrial Informatics* 19, 1 (2022), 286–295.

[2] Wenzhe Guo, Mohammed E Fouda, Ahmed M Eltawil, and Khaled Nabil Salama. 2021. Neural coding in spiking neural networks: A comparative study for robust neuromorphic systems. *Frontiers in Neuroscience* 15 (2021), 638474.

[3] Mark Horowitz. 2014. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. IEEE, 10–14.

[4] Seijoon Kim, Seongsik Park, Byunggook Na, and Sungroh Yoon. 2020. Spiking-yolo: spiking neural network for energy-efficient object detection. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 11270–11277.

[5] Youngeun Kim and Priyadarshini Panda. 2021. Revisiting batch normalization for training low-latency deep spiking neural networks from scratch. *Frontiers in neuroscience* (2021), 1638.

[6] Yuhang Li, Ruokai Yin, Hyoungseob Park, Youngeun Kim, and Priyadarshini Panda. 2022. Wearable-based Human Activity Recognition with Spatio-Temporal Spiking Neural Networks. *arXiv preprint arXiv:2212.02233* (2022).

[7] Lumin Liu, Jun Zhang, S.H. Song, and Khaled B. Letaief. 2020. Client-Edge-Cloud Hierarchical Federated Learning. In *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*. 1–6. https://doi.org/10.1109/ICC40277.2020.9148862

[8] Jesus L Lobo, Javier Del Ser, Albert Bifet, and Nikola Kasabov. 2020. Spiking neural networks and online learning: An overview and perspectives. *Neural Networks* 121 (2020), 88–100.

[9] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*. PMLR, 1273–1282.

[10] Nicolas Skatchkovsky, Hyeryung Jang, and Osvaldo Simeone. 2020. 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, 8524–8528.

[11] Yeshwanth Venkatesha, Youngeun Kim, Leandros Tassiulas, and Priyadarshini Panda. 2021. Federated learning with spiking neural networks. *IEEE Transactions on Signal Processing* 69 (2021), 6183–6194.

[12] Kan Xie, Zhe Zhang, Bo Li, Jiawen Kang, Dusit Niyato, Shengli Xie, and Yi Wu. 2022. Efficient federated learning with spike neural networks for traffic sign recognition. *IEEE Transactions on Vehicular Technology* 71, 9 (2022), 9980–9992.

[13] Zhanglu Yan, Jun Zhou, and Weng-Fai Wong. 2021. Energy efficient ECG classification with spiking neural network. *Biomedical Signal Processing and Control* 63 (2021), 102170.

[14] Helin Yang, Kwok-Yan Lam, Liang Xiao, Zehui Xiong, Hao Hu, Dusit Niyato, and H Vincent Poor. 2022. Lead federated neuromorphic learning for wireless edge artificial intelligence. *Nature communications* 13, 1 (2022), 4269.

[15] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. 2019. Bayesian nonparametric federated learning of neural networks. In *International conference on machine learning*. PMLR, 7252–7261.