Federal SNN Distillation: A Low-Communication-Cost Federated Learning Framework for Spiking Neural Networks

Federal SNN Distillation: A Low-Communication-Cost Federated Learning Framework for Spiking Neural Networks

Zhetong Liu[1, a], Qiugang Zhan[2, b], Xiurui Xie[3, c, *], Bingchao Wang[4, d], Guisong Liu[5, e, *]

[1]School of Computer Science, Zhongshan Institute, University of Electronic Science and Technology of China, Zhongshan, China

[2]School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China

[3]School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China

[4]School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China

[5]Economic Information Engineering School, Southwestern University of Finance and Economics, Chengdu, China

[a]q21330@qq.com, [b]202011081607@std.uestc.edu.cn, [c]xiexiurui@uestc.edu.cn, [d]1432678283@qq.com, [e]gliu@swufe.edu.cn

ABSTRACT: In recent years, research on the federated spiking neural network (SNN) framework has attracted increasing attention in the area of on-chip learning for embedded devices, because of its advantages of low power consumption and privacy security. Most of the existing federated SNN frameworks are based on the classical federated learning framework -- Federated Average (FedAvg) framework, where internal communication is achieved by exchanging network parameters or gradients. However, although these frameworks take a series of methods to reduce the communication cost, the communication of frameworks still increases with the scale of the backbone network. To solve the problem, we propose a new federated SNN framework, Federal SNN distillation (FSD), whose communication is independent of the scale of the network. Through the idea of knowledge distillation, FSD replaces the network parameters or gradients with the output spikes of SNN, which greatly reduces the communication while ensuring the effect. In addition, we propose a lossless compression algorithm to further compress the binary output spikes of SNN. The proposed framework FSD is compared with the existing FedAvg frameworks on MNIST, Fashion MNIST and CIFAR10 datasets. The experiment results demonstrate that FSD communication is decreased by 1-2 orders of magnitude when reaching the same accuracy.

1.Introduction

In today's Internet of Everything era, more and more mobile devices and embedded devices are connected to the Internet and become an integral part of artificial intelligence. However, as reflected in many cases of privacy leakage [1], the privacy issue of local data on mobile devices cannot be ignored [2] nowadays. In addition, the training of traditional neural networks is usually carried out on a dedicated

Graphics Processing Unit (GPU), which requires high energy consumption support. This is not friendly to mobile devices that are designed to be lightweight obviously.

Federated SNN [3] is proposed to solve the privacy and power consumption problems, which is a distributed training model implemented by spiking neural networks. In federated SNN, each device jointly trains a common SNN model without exchanging private data with other devices or servers. Federated Learning (FL) [4] provides a solution to the problem of privacy leakage based on communication with no data participation and encrypted transmission; Spiking neural network (SNN) [5] can reduce training power consumption by simulating potential changes and neural spikes of biological neurons and replacing real number outputs of traditional artificial neural network (ANN) with discrete binary sequences.

In distributed training scenarios where mobile and embedded devices are involved, the amount of traffic consumed by the device for communicating with the server is usually limited and expensive, such as mobile networks with billing by traffic [6]. However, FL mode requires frequent exchange of network training information between participating devices and central servers. The traditional distributed Stochastic Gradient Descent [7] (distributed SGD) algorithm switches the network gradients increment each round, and the communication of each round is proportional to the network parameters. Despite the use of SNN, it will still bring tens or even hundreds of gigabytes of communication overhead when the training network is large, which is unacceptable in federal SNN training.

To reduce communication costs and improve communication efficiency, a variety of improved methods for communication are proposed, such as Federal Average algorithm (FedAvg) [4], gradient quantization [8]-[9], and gradient sparsity [10], etc. Among them, the Federal Average algorithm has been widely cited and improved in the study of Federal SNN because of its excellent training effect. Although these methods can reduce the communication costs of distributed training to some extent, they are a kind of improvement of distributed SGD algorithm, for which the communications of the improved FL framework are still positively correlated with the network scale. This limits the further expansion of the network.

We propose a method to get rid of the influence of network scale on federated SNN communication. In order to achieve this, we design a scheme to apply a distillation method on federated SNN training, to reduce single-round communication. Specifically, the contributions of this paper are summarized as follows:

- We propose a Federal SNN Distillation (FSD) framework. Our framework uses output spikes instead of network parameters or gradients to participate in communication, significantly reducing the communication costs while maintaining the privacy and low power consumption advantages of federated SNN.
- We design a new loss function of the distillation train and a federated aggregation method. The loss function and aggregation scheme can be adapted to different SNN models, and is able to achieve a better training effect when combined with FSD.
- We design a lossless compression algorithm. According to the binary characteristic of SNN spikes, the algorithm nondestructively reduces the number of bits that communication information takes up by merging the spikes, and the network communication is further reduced.
- We conduct a large number of experiments on three data sets, to compare our framework with the Federated Averaging frameworks. The total communication of our algorithm is reduced by 1-2 orders of magnitude compared to Federated Average when achieving the same effect.

In the rest of this paper, we review the related works in section 2, i.e., spiking neural networks, federated SNN, and knowledge distillation. In section 3, we detail the proposed FSD framework and other work. In section 4, we conduct experiments and report experimental results to compare and analyze the communication costs of our framework and FedAvg frameworks. In section 5, we conclude the work of this paper.

2.Related Work

2.1.Spiking Neural Network

The spiking neural network has a different neuron model from the traditional neural network. Leaky-integrate-and-fire (LIF) [11] is a classical spiking neuron model, which has been widely used in the studies of SNN [12]-[14]. LIF abstracts the neurons as a resistor-capacitor (RC) circuit. The capacitor in the circuit will be charged by the input current, and when the capacitor voltage reaches a certain threshold, the current pulse will be released, and the voltage will return to zero. At rest, the capacitor also discharges slowly, and the rate of discharge decreases gradually with the voltage. LIF model describes the change of neuron membrane potential in the continuous-time domain, which is important in theoretical research. On computing devices, however, we need an explicit, discrete formula to achieve the implementability of the model. In [12], the differential equation is firstly solved by the Euler method and the iterative expression is obtained to realize the discretization of membrane potential. Then, the current function is simplified to the weighted sum of front neurons (or inputs) to obtain the discretization description of the current. The specific formula is:

$$u_i^{t+1,n+1} = ku_i^{t,n+1}\left(1 - o\left(u_i^{t,n+1} - u_{rest}\right)\right) + \sum_{j=1}^{l(n)} w_{ij}^n o\left(u_j^{t+1,n} - u_{rest}\right), \quad (1)$$

where u is the membrane potential, n and i denote the n-th layer and the i-th neuron of this layer respectively, t is time step, k is a constant, $o(\cdot)$ is the unit step function describing the spike output, which

$$o(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}, \quad (2)$$

and w is the connection weight of neurons. Equation (1) is a binary iterative sequence of n and t, which describes the layer-by-layer propagation of neuron spike signals over time in a very orderly manner, see **Figure 1**. The model used in [14] is a simplified version of (1), which ignores the sharp drop of the membrane potential after spike transmission, and the calculation of the spike only depends on the previous moment. Thus, it weakens the dependence of the membrane potential on the spike signal of the front layer. Meanwhile, [14] has also added Batch Normalization Through Time (BNTT) [15] technology on top of the basic model. Instead of using simple linearity as the activation function of weighted sum, it uses the statistics of weighted sum in the mini-batch of a time step and standardizes their calculations.

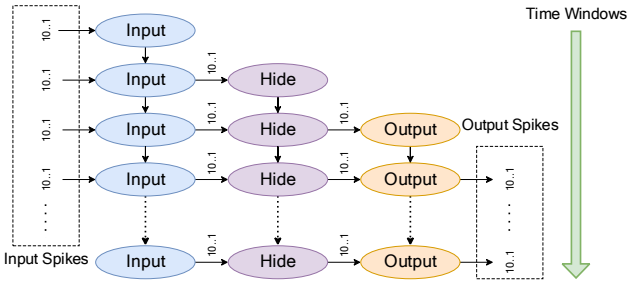


Figure 1: An implementation model of SNN that propagates layer by layer over time step. Longitudinal denotes discrete time steps; Transverse denotes structure of network, i.e., input layer, hidden layer (middle layer) and output layer.

There are many special studies on the training of SNN [16]-[20], while the neural network composed of LIF neurons can also be trained in backpropagation. However, the derivative of the unit step function describing the spike generation tends to infinity at zero, so it is necessary to find an approximate function to replace it. [12] chooses rectangular function as a substitute derivative function, the narrower the

rectangle, the closer it is to the shape of the original derivative; [14] uses a piece-wise linear approximation to achieve this function.

In our work, the proposed framework is generic and does not depend on a particular SNN implementation. We will conduct experimental tests on models of [12] and [14] to show good results in different SNN base models.

2.2.Federated SNN

To protect the privacy and take full advantage of scattered data on embedded devices used SNN, federated SNN is proposed. It adds resource-constrained devices into federated learning and overcomes the problem of insufficient training data. Federated SNN integrates SNN models on distributed devices into federated learning framework and exploits the advantages of SNN in the case of limited training data set. The design of the federal SNN program focuses on the structure of the SNN model and the global network parameters update program, which have a decisive influence on the training effect. The basic structure of federated SNN is shown in **Figure 2**.

For SNN and its training, Nicolas et al. [3] proposed FL-SNN with probabilistic SNN model and log-likelihood function as loss function in local training. Yeshwanth et al. [14] proposed a federal learning framework in a realistic federal learning scenario. In the previous section of **Spiking Neural Network**, we introduce their neuron model in detail, and their SNN adopted VGG9 [21] as the topological structure. For communication, [3] considering the problem that traditional federated learning regularly produces much communication consumption, synapse pruning operation is added into their algorithm to reduce communication; [14] focuses on the energy consumption of network training and does not discuss the communication consumption of federated learning.

Both [3] and [14] adopt Federated Average algorithm (FedAvg) [4] as the global network parameters updating scheme. FedAvg is a classical algorithm in federated learning for model aggregation. A typical federated learning system that applies the FedAvg has a server (or base station, central node) and several clients. The client maintains its own private data set, performs local training on it, and communicates with the client regularly to participate in global training with parameters exchange. The communication of this federated learning system depends on the size of the model being trained, so the communication cost is unacceptable when applied to complex networks such as ResNet-50 [22]. Some studies attempt to reduce the single-round communication by model compression, such as quantization scheme [8]-[9], gradient sparse and gradient compression schemes [10], and other common model compression algorithms [23]-[25]. However, these solutions can only alleviate the transmission bottleneck, for the transmission is still positively correlated with the network size. After the network scale expands further, the compression space of transmission information will gradually saturate.

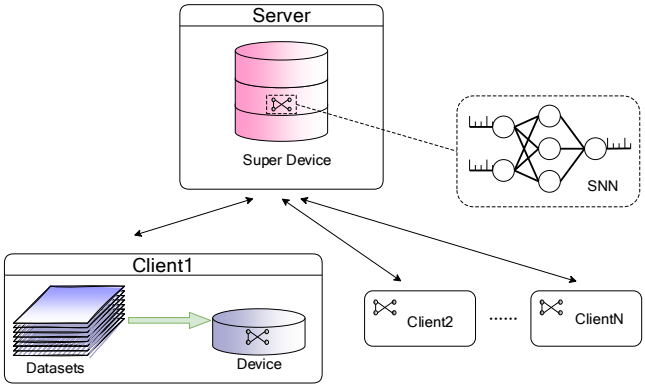


Figure 2: The schematic diagram of federated SNN. The training data is stored locally in the client. A server collaborates with multiple clients to train one SNN.

Strictly speaking, *federated average* refers only to the federated learning algorithm on traditional deep neural networks. However, the federated learning framework improved on it has a common feature, that is, calculating the average after transferring parameters or gradients. For simplicity, *federated*

average (FedAvg) in the rest of this paper is used to refer to federated learning frameworks that use federated average algorithms and communicate with parameters or gradients, whether used in SNN or ANN.

2.3.Knowledge Distillation

Knowledge distillation (KD) [26] is an extraction scheme for network knowledge, which can transfer the knowledge of a trained large neural network to a small network, making the small network show a very similar effect to the large network. Small-scale networks have advantages such as fast forward computing speed and small resource consumption. In KD, the network of acquiring knowledge is called the student network, and the network of transferring knowledge is called the teacher network.

The idea behind the classical KD algorithm is that the student network and the teacher network share a dataset (also called Book in some studies) on which the student network trains. Different from the traditional training, the loss function of the student network in the distillation training not only uses the label (hard-label) of the data itself, but also needs to add the distillation items composed of soft-max labels (soft-label [27], or called logit [26]) output by the teacher network. They are combined to carry out backpropagation according to a certain weight.

KD proves that the knowledge of the network not only exists in parameters but also can be reflected through output. Inspired by this, ANN researchers proposed the federal distillation (FD) framework [28]-[30], which communicates based on output instead of the gradients. This also provides ideas for our work.

3.Federal SNN Distillation

In this section, we introduce the proposed Federated SNN Distillation (FSD) framework. We first describe the basic structure of the framework and then explain in detail the polymerization scheme, distillation loss function, and lossless compression algorithm used in the framework. Finally, we demonstrate our framework in pseudo-code form in **Algorithm 1**.

3.1.Federated SNN Structure

Federated SNN distillation (FSD) runs on a typical federated learning system consisting of one server and N clients, each of the clients $i = 1,2, \dots N$ holds a private data set D_i. Before the training begins, the client and the server conduct initial communication to exchange status words, and then perform initialization respectively: the server downloads the preset public dataset X_p; The clients download the public data set X_p and divides their private dataset D_{iT} and validation set D_{iV}. This process is visually shown in **Figure 3**. Next, we will describe an iteration on the client and the server respectively.

1) Client: The clients first initialize SNN. Then, clients download the latest compressed global spike tensor from the server and decompresses it to obtain S_g for distillation training on X_p. After distillation, clients train SNN with pre-knowledge locally on D_{iT}, and the training scheme depends on the basic SNN model used. The clients complete the local update for this round after the training. Next, the clients test the updated SNN to on D_{iV} and forward propagates it on X_p to get the local accuracy acc_i and the local spike tensor S_i. After S_i is compressed and get ${S_i}^c$, it will be uploaded to the server with acc_i, and then the client can wait for the next round of updates.

2) Server: the server receives all the updates ${S_i}^c$ uploaded by the client, decompresses them to get S_i, and then aggregates them to get a new S_g. Like the client, the server SNN performs distillation training on the aggregated S_g to obtain model update. It is worth to note that the server SNN is not initialized before the train, so the updated network has accumulated knowledge of previous rounds. The aggregated S_g loses the binary property and cannot be compressed and transferred directly, so we adopt a voting scheme, the S_g used for the transfer will be voted by the S_i uploaded from the clients (in fact, this is equivalent to rounding operation, so we actually use the *Round* function to implement it). Finally, the processed S_g is compressed, and a complete round of training is completed.

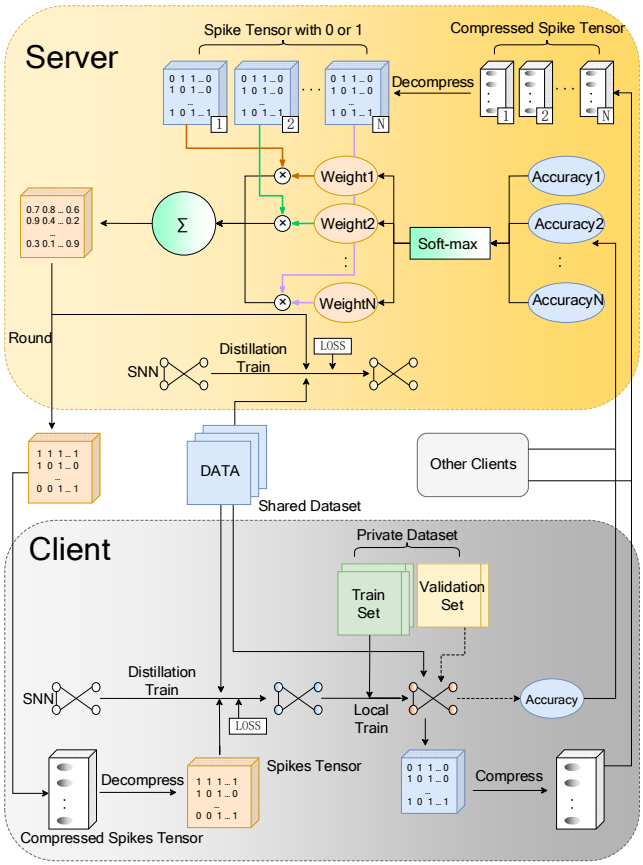


Figure 3: Federal SNN Distillation. The figure shows the complete flow of FSD, as well as the data transmission route. The upper part of the figure depicts the process of aggregation and distillation on the server, and the lower part depicts the process of training and distillation on the client.

3.2.Federal Aggregate Scheme

We aggregate the spikes tensor uploaded by the client by weighting the accuracy. The private dataset of each client is divided into the training set D_{iT} and validation set D_{iV}. The training set is used for local training of the client network, and the validation set is used for testing local accuracy on the client network after training. The client uploads the local accuracy rate along with the spikes tensor, and the server soft-max these local accuracy rates to get the aggregation weights and aggregation. The **Merge** function in Algorithm 1 does the job.

$$S_g = \sum_{i=1}^{N} softmax(\{acc_i | i = 1,2,\dots,N\})_i S_i \tag{3}$$

We note that the elements in the local spike tensor S_i have only two values, while the elements in the aggregate global spike tensor S_g will be floating-point numbers in the range [0,1]. Thus, it does not exactly match the output format of SNN. In the following work, we will solve this problem.

3.3.SNN Distillation

The key problem of distillation is the training loss function, which has a direct influence on the distillation effect. As we know, the SNN technology is not as mature as ANN, so only one distillation scheme [31] of SNN direct training using three loss functions is proposed for the first time. The scheme of [31] is suitable for distillation of local SNN with large size of time windows, while in the application scenario of federated SNN, the size of SNN time windows is usually limited by computing resources. To achieve federated SNN distillation and weaken the constraint of the training algorithm on SNN time

windows, we propose a new loss function strategy. It is implemented by the **Distillation** function in **Algorithm 1.**

Our loss function L consists of two terms L_T and λL_F, i.e.,

$$L = L_T + \lambda L_F \quad .(4)$$

The first refers to the distillation method of spike tensor in [31], for that we hope that the output spike tensor of student SNN is as close as possible to any position of teacher SNN, so as to learn the knowledge of teacher network more completely. The second is a palliative of the first term, which allows the shape of the spike output of the student network to be no longer exactly the same as that of the teacher network, but only the consistency of the spike frequency. Through the preset parameters of this term, we can adjust the degree of matching between two spike matrixes. Considering the value of the frequency vector, we describe it by the cross-entropy loss. The specific formula is as follows.

$$L_T = \frac{1}{|C|T_W}\sum_{c=1}^{|C|}\sum_{t=1}^{T_W}(s_{ct} - \hat{s}_{ct})^2 \quad (5)$$

$$L_F = -\sum_{c=1}^{|C|}\hat{p}_c \log p_c \quad (6)$$

$$p_c = \frac{1}{T_W}\sum_{t=1}^{T_W} s_{ct} \quad (7)$$

Where $|C|$ is the number of classes of training data, T_W is the size of SNN time windows; s_{ct} and $\hat{s}_{ct}$ are values in the mini-batch predicted spike matrix and target spike matrix respectively. p_c and $\hat{p}_c$ are the frequency vectors, which can be calculated by the spike matrix. The firing frequency of SNN spike to this type is described in Formula (6); λ is a preset constant describing the extent to which the spike frequency affects the loss function.

3.4.Compression for Spike Tensor

Relative to ANN, the SNN output spikes train adds a time dimension, and direct transmission will add additional communication. However, we note that due to the binary nature of the Spikes train, elements in the spike tensor can be expressed lossless with only one bit of precision. Inspired by this, we propose a spikes tensor-compression algorithm based on the binary conversion operation, which eliminates the extra dimension of the spike tensor. **Algorithm 2** describes the implementation details of compression, which corresponds to **Compress** and **Decompress** in **Algorithm 1.**

Algorithm 1 Federated Spiking Neural Network Distillation

Input: Number of clients N, private dataset $D_i = (D_{iT}, D_{iV})$, public dataset (X_p), time window's size T_W, learning rate α, global epochs E_g, local epochs E_l.

For $e = 1, 2, \ldots, E_g$:
  Client i parallel do:
    reset SNN;
    download ${S_g}^c$ from server;
    S_g: = **Decompress** $({S_g}^c)$
    Distillation (SNN, X_p, S_g);
    Train (SNN, D_{iT}, α, E_l);
    S_i: = SNN (X_p);
    ${S_i}^c$: = **Compress** (S_i)
    acc_i: = Test (SNN, D_{iV});

11. upload $S_i{}^c$ and acc_i to server;
12. Server:
13. For $i = 1, 2, \ldots, N$:
14. S_i: = **Decompress** ($S_i{}^c$)
15. S_g: = **Merge** ($\{S_i | i = 1,2, \ldots, N\}$, $\{acc_i | i = 1,2, \ldots, N\}$)
16. **Distillation** (SNN, X_p, S_g);
17. S_g: = Round (S_g)
18. $S_g{}^c$: = **Compress** (S_g)
19. return SNN

Note: Functions **Merge**, **Distillation**, and **Compress** (**Decompress**) are explained in detail in sections 3.2, 3.3, and 3.4 respectively.

Our compression algorithm ensures that the compressed tensor elements are still low-precision integers. As a result, they do not require floating point storage space like ANN, further reducing communication costs.

In general, we add two schemes to reduce the communication costs of federated SNN. First, we transform the communication body into spike tensor that occupies less space, so that the communication does not increase with the growth of model size. Then, we design a lossless compression algorithm for the spike tensor, which reduces the number of bits stored and further reduces the number of bytes communicated. At the same time, in order to match the new framework, we design a new aggregation method and loss function for distillation. The collaborative work of these programs constitutes the main part of the FSD, resulting in a new federal SNN framework fundamentally different from the traditional structure.

Algorithm 2 Compress and Decompress

Compress ($S: Z^{c \times t};\ \forall i, j, s_{ij} = 0\ or\ 1$):
1. $SC \coloneqq \mathbf{0};\ SC: Z^c$
2. For $i = 1, 2, \ldots, c$
3. for $j = 1, 2, \ldots, t$:
4. $sc_i \coloneqq sc_i \times 2 + s_{ij}$;
5. return SC

Decompress ($SC: Z^t$):
1. $S \coloneqq \mathbf{0}; S: Z^{c \times t}$
2. For $i = 1, 2, \ldots, c$:
3. $sc \coloneqq sc_i$;
4. for j = $t, t-1, \ldots, 1$:
5. $s_{ij} \coloneqq sc \bmod 2$;
6. $sc \coloneqq sc \operatorname{div} 2$;
7. return S

4.Experimentation and Results

4.1.Experimental Setup

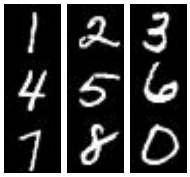
4.1.1.Datasets and Network Structures. We evaluate FSD framework on image classification task. We take four datasets: MNIST [32], fashion-MNIST [33], EMNIST [34] and CIFAR10 [35], as shown in **Figure 4**. Among them, CIFAR10 is a color image classification dataset, and MNIST is a classical

handwritten digital classification dataset. The Fashion-MNIST dataset contains different types of clothing commodities, EMNIST expands handwritten letters based on handwritten numbers. Fashion-MNIST and EMNIST are completely consistent with the image size of MNIST, which is the adaptation and expansion of MNIST. For MNIST and EMNIST datasets, we adopt the same setup as in [30]: taking MNIST as the clients' private dataset and EMNIST as the public distillation dataset. On Fashion-MNIST and CIFAR10, we randomly select 8000 samples as the public dataset, and randomly select 10000 samples for each client as the private dataset.

4.1.2.Federated Training Configuration. We use one server with four clients participating as the basic structure for federated training. Before the training, 10% of the client's private dataset is used as the validation set. The federated global training is set to run in 50 global epochs, with one device first connecting to the server and training, and all four devices participating in the second epoch. In each global epoch, the clients carry out 5 epochs of distillation and 5 local epochs of training, and the server will carry out 5 epochs of distillation. After completion, we record the total communication traffic and test accuracy at the end of the global epoch for comparative analysis. The FedAvg for comparison also uses this configuration.

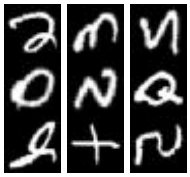
4.1.3.Optimization Algorithms and Hyperparameters. In our experimental setup, both distillation training and local training use the currently popular Adam optimizer [36], which is an extension of the traditional SGD algorithm and has many excellent features. We set a learning rate of 1×10^-5 in both distillation training and local training, which has shown good results in previous practice tests. The spike threshold $\theta = 0.25$, the decay factor $\tau = 0.1$ and the size of time window $T_W = 8$ are set for the specific hyperparameters of SNN. We apply identical optimization algorithms and hyperparameters to the FedAvg models of the control group.

4.1.4.Baseline. We conduct two groups of comparative experiments. The first group of experiments uses the SNN model in [12], which is the most popular SNN model, and we named it DTSNN in our paper. The second group of experiments, use the same SNN model as Venkatesha Y et al. [14] achieved the SOTA, named V+FedAvg in our paper. We use the same SNN model to implement the FSD framework, denoted as V+FSD. DTSNN+FedAvg and V+FedAvg provide a baseline of upstream communication when target accuracy is achieved. Comparison of the experimental results and visual presentation is presented in next section.

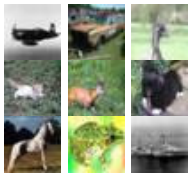


(a) MNIST

(b) Fashion-MNIST



(c) EMNIST

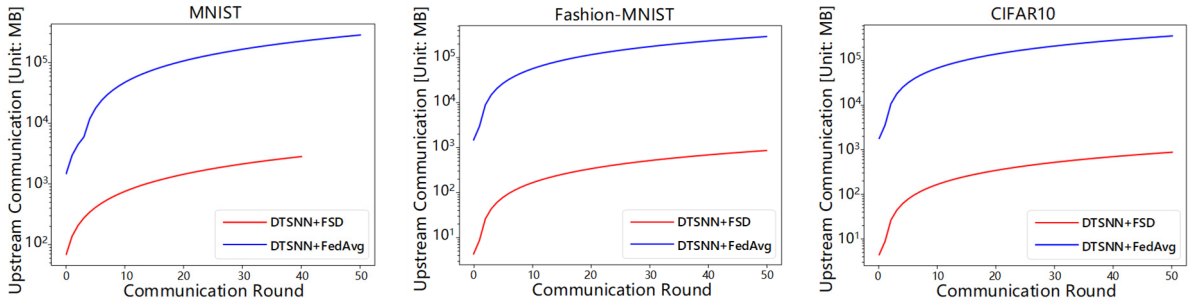


(d) CIFAR10

Figure 4: Samples of datasets.

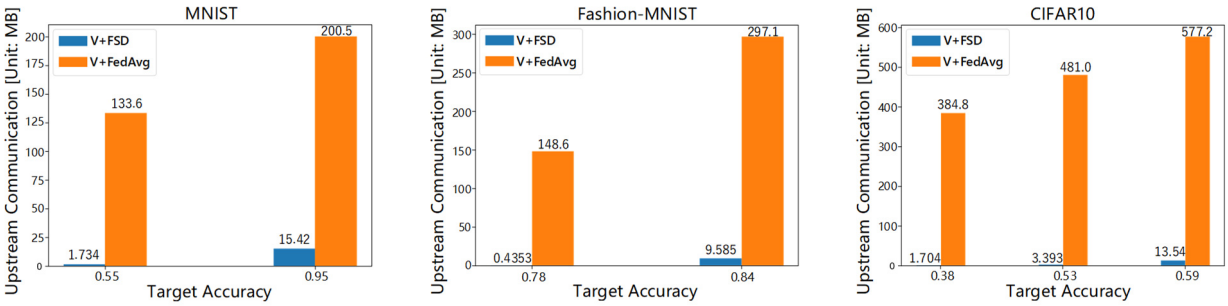
4.2.Experimental Results and Analyses

4.2.1.Experiment on DTSNN. **Table 1** shows the experimental results using SNN of DTSNN [12] when reaching the same target accuracy. The content under DTSNN+FSD and DTSNN+FedAvg items is the number of communication bits, in MB, with four significant digits reserved. Item Times shows the ratio of DTSNN+FedAvg to DTSNN+FSD communication bits, reserving three significant digits. On the three datasets, the communication of DTSNN+FSD is decreased compared with that of DTSNN+FedAvg, with the highest reaching 405 times. On MNIST, it takes 27.56MB of communication for DTSNN+FSD to reach 97% accuracy, while DTSNN+FedAvg costs 148.6MB. Our method saves 5.39 times of communication consumption. When target accuracy is increased from 97% to 98%, the increased communication consumption of DTSNN+FSD is slightly more than DTSNN+FedAvg, but it is still 1.79 times less overall. On Fashion-MNIST and CIFAR10, DTSNN+FSD reduces communication by 31 and 25 times with 84% and 38% target accuracy.



(a) MNIST (b) Fashion-MNIST (c) CIFAR10

Figure 5: Graph of upstream communication bits per round for DTSNN+FSD and DTSNN+FedAvg. The abscissa is the communication round, and the logarithmic ordinate is the number of communication bits, in MB.



(a) MNIST (b) Fashion-MNIST (c) CIFAR10

Figure 6: Communication bits comparison diagram under target accuracy. When the target accuracy is low, the columns representing V-FSD is extremely low.

We also draw the upstream communication consumption curve by rounds (each global epoch contains a round of communication), as shown in **Figure 5**. It shows that the overall communication consumption of DTSNN+FSD is lower than that of DTSNN+FedAvg on three datasets. In addition, **Figure 5** takes logarithmic coordinates as the vertical axis, but the curve shapes of DTSNN+FedAvg and DTSNN+FSD are almost the same. This proves that the communication cost of FSD is lower while ensuring a low growth rate, and FSD is capable of maintaining its low communication advantage even in the task requiring multiple rounds of communication.

Table 1: Results of FSD framework using SNN of DTSNN [12].

Dataset	Target-Acc (%)	Communication (MB)		Time s
		DTSNN+FSD	DTSNN+FedAvg	
MNIST	97	27.56	148.6	5.39
	98	165.6	297.1	1.79
F-MNIST	82	0.873	148.6	170.
	84	9.585	297.1	31.0
CIFAR10	32	0.887	360.0	406.
	36	13.32	1080.	81.1
	38	72.02	1800.	25.0

4.2.2.Experiment on V-SNN. **Table 2** shows the experimental results using the SNN of Venkatesha et al. [14]. V+FSD still consumes less communication cost than V+FedAvg. Compared with DTSNN+FSD, V+FSD using SNN of [14] achieves higher target accuracy in CIFAR10, while the effect is similar in MNIST and Fashion-MNIST datasets. On MNIST, to achieve 95% accuracy, FSD requires 15.42MB of communication, while V+FedAvg costs 200.5MB. FSD reduces communication consumption by 12 times. At 84% accuracy on Fashion-MNIST, communication of V+FedAvg is 31 times higher than that of V+FSD. At 59% accuracy on CIFAR10, V+FedAvg is 42 times higher than V+FSD in communication. It can be seen that different SNN basic frameworks have a certain impact on the training effect.

Table 2: Results of FSD framework using SNN of Venkatesha et al. [14].

Dataset	Target-Acc (%)	Communication (MB)		Time s
		V+FSD	V+FedAvg	
MNIST	55	1.734	133.6	77.0
	95	15.42	200.5	13.0
F-MNIST	78	0.4353	148.6	341.
	84	9.585	297.1	31.0
CIFAR10	38	1.704	384.8	226.
	53	3.393	481.0	142.
	59	13.54	577.2	42.6

To display the saved communication more intuitively, we drew a bar chart of communication bits under the target accuracy in this part of the experiment, as shown in **Figure 6**. There is an obvious difference in the height of the two frames. The lower the target accuracy is, the more obvious the difference is. **Figure 6** shows that FSD has natural advantages in scenarios where accuracy requirements are not very strict and the energy consumption is limited.

5.Conclusion

In this paper, we study the communication characteristics of federated SNN under the classical Federated Average framework, and discuss the problem of high communication cost that commonly exists in the federated SNN under the current FedAvg framework. Then inspired by the Knowledge Distillation algorithm, we propose FSD, a new training framework for federated SNN. We carry out experiments on the FSD framework in the image classification task and prove that FSD can reduce the total communication compared with FedAvg when reaching the target accuracy. As we can see, the decrease can reach 1-2 orders of magnitude. This saves communication costs greatly and makes it feasible to deploy Federated Learning in an environment with limited device power consumption and communication bandwidth. We believe that in the current state of ai development, FSD can have its role on a large number of occasions.

However, as with any framework, there're environments in that FSD is not suitable for use. FSD uses relatively complex calculations for the loss function, which raises the threshold of the computing capability of the device. Therefore, in a specific application scenario, the most appropriate scheme needs to be considered more carefully.

Federated SNN and distillation learning are relatively new technologies in the field of artificial intelligence, and there are many aspects worthy of further study. So far, due to the characteristics of SNN and Knowledge Distillation, it is difficult to theoretically guarantee the convergence of the scheme and the robustness of the scheme in more complex distributed scenarios (such as non-IID data, an unstable connection of communication equipment, or timeout problems). In the future, we will carry out further research on these issues to achieve the enhancement and perfection of FSD.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China under Grant 61806040 and 61771098, the Natural Science Foundation of Guangdong Province under Grant 2021A1515011866 and Sichuan Province under Grant 2021YFG0018, and the Social Foundation of Zhongshan Sci-Tech Institute under Grant 420S36.

REFERENCES

[1] Krishnamurthy B, Naryshkin K, Wills C. Privacy leakage vs. protection measures: the growing disconnect[C]//Proceedings of the Web. 2011, 2(2011): 1-10.
[2] Krishnamurthy B, Wills C E. Privacy leakage in mobile online social networks[M]//Proceedings of the 3rd Wonference on Online social networks. 2010: 4-4.
[3] Skatchkovsky N, Jang H, Simeone O. Federated neuromorphic learning of spiking neural networks for low-power edge intelligence[C]//ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2020: 8524-8528.
[4] McMahan B, Moore E, Ramage D, et al. Communication-efficient learning of deep networks from decentralized data[C]//Artificial intelligence and statistics. PMLR, 2017: 1273-1282.
[5] Ghosh-Dastidar S, Adeli H. Spiking neural networks[J]. International journal of neural systems, 2009, 19(04): 295-308.
[6] Ibrahim H A, Nossier B M, Darwish M G. Billing system for Internet service provider (ISP)[C]//11th IEEE Mediterranean Electrotechnical Conference (IEEE Cat. No. 02CH37379). IEEE, 2002: 260-268a.
[7] Gemulla R, Nijkamp E, Haas P J, et al. Large-scale matrix factorization with distributed stochastic gradient descent[C]//Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. 2011: 69-77.
[8] Alistarh D, Grubic D, Li J, et al. QSGD: Communication-efficient SGD via gradient quantization and encoding[J]. Advances in Neural Information Processing Systems, 2017, 30: 1709-1720.
[9] He Y, Zenk M, Fritz M. CosSGD: Nonlinear Quantization for Communication-efficient Federated Learning[J]. arXiv preprint arXiv:2012.08241, 2020.
[10] Sattler F, Wiedemann S, Müller K R, et al. Robust and communication-efficient federated learning from non-iid data[J]. IEEE transactions on neural networks and learning systems, 2019, 31(9): 3400-3413.
[11] Burkitt A N. A review of the integrate-and-fire neuron model: I. Homogeneous synaptic input[J]. Biological cybernetics, 2006, 95(1): 1- 19.
[12] Wu Y, Deng L, Li G, et al. Direct training for spiking neural networks: Faster, larger, better[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2019, 33(01): 1311-1318.
[13] N. Rathi, G. Srinivasan, P. Panda, and K. Roy, "Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation," 2020.
[14] Venkatesha Y, Kim Y, Tassiulas L, et al. Federated Learning with Spiking Neural Networks[J]. arXiv preprint arXiv:2106.06579, 2021.

[15] Y. Kim and P. Panda, "Revisiting batch normalization for training low latency deep spiking neural networks from scratch," 2020.
[16] Xie X, Qu H, Liu G, et al. An efficient supervised training algorithm for multilayer spiking neural networks[J]. PloS one, 2016, 11(4): e0150329.
[17] Xie X, Qu H, Yi Z, et al. Efficient training of supervised spiking neural network via accurate synaptic-efficiency adjustment method[J]. IEEE transactions on neural networks and learning systems, 2016, 28(6): 1411-1424.
[18] Ankur Gupta and Lyle N Long, "Hebbian learning with winner take all for spiking neural networks," in 2009 International Joint Conference on Neural Networks. IEEE, 2009, pp. 1054–1060.
[19] Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer, "Training deep spiking neural networks using backpropagation," Frontiers in neuroscience, vol. 10, pp. 508, 2016.
[20] Shihui Yin, Shreyas K Venkataramanaiah, Gregory K Chen, Ram Krishnamurthy, Yu Cao, Chaitali Chakrabarti, and Jae-sun Seo, "Algorithm and hardware design of discrete-time spiking neural networks based on back propagation with binary activations," in 2017 IEEE Biomedical Circuits and Systems Conference (BioCAS). IEEE, 2017, pp. 1–5.
[21] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.
[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.
[23] Konečný J, McMahan H B, Yu F X, et al. Federated learning: Strategies for improving communication efficiency[J]. arXiv preprint arXiv:1610.05492, 2016.
[24] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in Advances in Neural Information Processing Systems (NeurIPS), vol. 2, 1990, pp. 598–605.
[25] S. Wiedemann, H. Kirchhoffer, S. Matlage, P. Haase, A. Marbán, T. Marinc, D. Neumann, T. Nguyen, H. Schwarz, T. Wiegand, D. Marpe, and W. Samek, "DeepCABAC: A universal compression algorithm for deep neural networks," IEEE J. Sel. Top. Signal Process., vol. 14, no. 4, pp. 700–714, 2020.
[26] Hinton G, Vinyals O, Dean J. Distilling the knowledge in a neural network[J]. arXiv preprint arXiv:1503.02531, 2015.
[27] Polino A, Pascanu R, Alistarh D. Model compression via distillation and quantization[J]. arXiv preprint arXiv:1802.05668, 2018.
[28] Jeong E, Oh S, Kim H, et al. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data[J]. arXiv preprint arXiv:1811.11479, 2018.
[29] Itahara S, Nishio T, Koda Y, et al. Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-iid private data[J]. arXiv preprint arXiv:2008.06180, 2020.
[30] Sattler F, Marban A, Rischke R, et al. Communication-efficient federated distillation[J]. arXiv preprint arXiv:2012.00632, 2020.
[31] Kushawaha R K, Kumar S, Banerjee B, et al. Distilling Spikes: Knowledge Distillation in Spiking Neural Networks[C]//2020 25th International Conference on Pattern Recognition (ICPR). IEEE, 2021: 4536-4543.
[32] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE*, 86(11):2278-2324, November 1998.
[33] Xiao H, Rasul K, Vollgraf R. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms[J]. 2017.
[34] Cohen, S. Afshar, J. Tapson, and A. van Schaik, "EMNIST: An extension of mnist to handwritten letters," arXiv preprint arXiv:1702.05373, 2017.
[35] Krizhevsky A, Hinton G. Learning multiple layers of features from tiny images[J]. 2009.

[36] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” arXiv preprint arXiv:1412.6980, 2014.