

Article

PriSpark: Privacy-Preserving and Communication-Efficient Federated Spiking Neural Learning via Event-Sparse Adaptive Aggregation

Xin Liu ^{1,2}, Honglei Yao ^{1,2}, Shanjie Xu ^{1,2}, Yuhui Jin ^{1,2}, Zijie Pan ³ , Jiamin Zheng ⁴  and Li Tan ^{4,*} 

¹ State Grid Shandong Electric Power Company, Jinan 250001, China; liuxin.shandonggrid@gmail.com (X.L.); yaohonglei1118@163.com (H.Y.); shanjie1997@gmail.com (S.X.); jinyuhui2980@gmail.com (Y.J.)

² Shandong Key Laboratory of Energy Industry Internet Big Data Technology, Jinan 250003, China

³ Faculty of Data Science, City University of Macau, Macau SAR, China; zijiepan20@gmail.com

⁴ School of Computer Science and Artificial Intelligence, Beijing Technology and Business University, Beijing 100048, China; zhengjm@btbu.edu.cn

* Correspondence: tanli@th.btbu.edu.cn

Abstract

Federated learning enables collaborative model training without centralizing private data, yet its deployment on edge devices remains constrained by communication overhead, heterogeneous data distributions, and privacy leakage from shared model updates. Spiking neural networks offer an energy-efficient alternative to conventional artificial neural networks by transmitting sparse binary events rather than dense activations, but existing federated spiking learning methods still suffer from inefficient gradient exchange and insufficient privacy protection under non-independent and identically distributed data. This paper proposes PriSpark, a privacy-preserving and communication-efficient federated spiking neural learning framework that jointly exploits temporal event sparsity, adaptive client-side spike-gradient compression, and privacy-calibrated aggregation. The core of PriSpark is a novel Event-Sparse Differentially Private Federated Spiking Optimization algorithm, which converts local surrogate gradients into spike-saliency-aware sparse updates, dynamically allocates communication budgets across layers and time steps, and injects calibrated Gaussian noise after clipping in a low-dimensional event subspace. To mitigate accuracy degradation caused by aggressive compression and privacy perturbation, PriSpark further introduces a membrane-aware error-feedback mechanism and a heterogeneity-adaptive server aggregation rule that weights client updates according to spike activity stability and local distribution drift. Experiments on neuromorphic and vision benchmarks, including N-MNIST, DVS128 Gesture, CIFAR-10, and Fashion-MNIST, show that PriSpark achieves competitive or superior accuracy compared with federated artificial neural and spiking baselines while substantially reducing uplink communication. Under strong privacy constraints, PriSpark maintains stable convergence and improves the accuracy–communication–privacy trade-off, demonstrating its potential for privacy-sensitive edge intelligence with event-driven neural computation.

Keywords: federated learning; spiking neural networks; differential privacy; communication efficiency; event-sparse optimization; edge intelligence; neuromorphic computing



Academic Editors: Irfan Awan, Amna Qureshi and Muhammad Shahwaiz Afaqui

Received: 23 June 2026

Revised: 10 July 2026

Accepted: 16 July 2026

Published: 27 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Federated learning enables multiple clients to collaboratively train a model without uploading their raw data to a central server, and it has become a representative paradigm

for privacy-aware edge intelligence [1]. In a standard federated system, the server broadcasts a global model, selected clients perform local optimization, and the server aggregates the returned updates. This workflow reduces direct data exposure, but it does not fully solve privacy or efficiency problems. Shared model updates can still leak sensitive information through gradient inversion or membership inference attacks [2,3], and dense uplink communication becomes expensive when clients are bandwidth-limited mobile, wearable, or neuromorphic devices. These challenges become more severe under non-independent and identically distributed data, where local objectives $F_k(\theta)$ may deviate substantially from the global objective $F(\theta)$ and cause client drift [4–6].

Spiking neural networks provide a promising computational substrate for communication- and energy-constrained federated learning. Unlike conventional artificial neural networks that propagate dense real-valued activations, spiking neural networks represent information through temporally sparse binary events and membrane-potential dynamics. This event-driven representation is naturally compatible with low-power edge devices and neuromorphic sensors, as demonstrated by peer-reviewed work on surrogate-gradient learning, temporal backpropagation, learnable membrane dynamics, and neuromorphic hardware [7–14]. However, directly combining federated learning with spiking models still inherits the limitations of conventional federated optimization and compressed learning [4–6,15–17]: existing protocols typically transmit dense updates, use magnitude-based sparsification without considering spike dynamics, or lack formal privacy protection for transmitted update values and sparse coordinate patterns.

The central difficulty is that privacy, communication, and temporal spiking dynamics are tightly coupled rather than independent. Differential privacy protects model updates by clipping and perturbing them with calibrated noise [18,19], but dense perturbation injects noise into all d parameters even though many coordinates may be inactive or uninformative in event-driven computation. Sparse communication can reduce uplink cost, but a sparse mask m_k^r selected from local gradients may itself reveal private information, and magnitude-only sparsification may discard parameters that are important for future spike generation. Heterogeneous aggregation methods can reduce client drift, but conventional weights based mainly on data size do not reflect whether a client's spike activity is stable or whether its protected sparse update is dominated by privacy noise. These issues suggest that federated spiking learning requires an integrated mechanism in which the selected subspace Ω_k^r , the released update $\hat{\Delta}_k^r$, and the aggregation weight ω_k^r are designed together.

This paper proposes PriSparK, a privacy-preserving and communication-efficient federated spiking neural learning framework. In each communication round r , client k trains a local spiking model using surrogate gradients and obtains a local update Δ_k^r . Instead of transmitting this update densely, PriSparK computes a spike-gradient saliency score $g_{k,j}^r$ for each parameter coordinate j by combining update magnitude, presynaptic spike activity, postsynaptic surrogate sensitivity, and temporal spike stability. The client then constructs an event-sparse mask m_k^r , forms a corrected sparse update with membrane-aware error feedback, and releases a privacy-calibrated quantized update $\hat{\Delta}_k^r$. The server aggregates the received updates using a heterogeneity-adaptive coefficient ω_k^r that starts from the selected-client data proportion $\bar{p}_k$ and adjusts it according to bounded spike-stability and update-reliability statistics.

The novelty of PriSparK lies in its joint treatment of event sparsity, differential privacy, and federated aggregation. First, the proposed event-sparse compression rule selects coordinates according to their influence on spike-driven computation rather than gradient magnitude alone. A parameter is prioritized when it is connected to active presynaptic spikes and sensitive postsynaptic membrane states, and its layer has stable temporal firing behavior. Second, PriSparK applies privacy perturbation after sparse projection and

accounts for the information contained in both update values and sparse masks. This design avoids the utility loss caused by perturbing inactive coordinates while preventing the sparse index set from becoming an unprotected side channel. Third, the server uses spike-stability-aware aggregation to reduce the influence of unstable or noise-dominated client updates without accessing raw data, spike trains, membrane traces, or dense local gradients.

We evaluate PriSparK on four public benchmarks, including N-MNIST, DVS128 Gesture, CIFAR-10, and Fashion-MNIST. N-MNIST and DVS128 Gesture represent native event-stream learning, while CIFAR-10 and Fashion-MNIST are converted into spike trains to test the generality of the method on static visual data. Under a Dirichlet non-IID partition with concentration $\alpha_D = 0.3$, PriSparK achieves stronger accuracy than differentially private dense and sparse federated SNN baselines while reducing the normalized uplink communication to approximately $0.082 \times$ of dense 32-bit FedAvg-SNN. The improvements are especially clear on CIFAR-10 and DVS128 Gesture, where client heterogeneity and temporal spike dynamics are more challenging. Ablation studies further show that event-gradient saliency, adaptive compression, membrane-aware error feedback, sparse privacy perturbation, and heterogeneity-adaptive aggregation all contribute to the final performance.

The main contributions of this work are threefold. We introduce an event-sparse federated spiking optimization framework that aligns communication with spike-driven computation through the saliency score $g_{k,j}^r$ and the adaptive budget q_k^r . We develop a privacy-calibrated sparse release mechanism that clips, perturbs, and quantizes the selected update subspace while controlling leakage from sparse masks and bounded metadata. We design a heterogeneity-adaptive aggregation rule that weights protected sparse updates by data proportion, spike-activity stability, robust update-norm consistency, and privacy-noise awareness. Together, these components allow PriSparK to improve the accuracy–communication–privacy trade-off in federated spiking neural learning.

2. Related Work

2.1. Federated Learning, Privacy, and Communication Efficiency

Federated learning trains a shared model from decentralized client data without centralizing raw samples. FedAvg remains the canonical protocol, and subsequent analyses and systems have shown that statistical and system heterogeneity require explicit correction mechanisms [1,15]. FedProx adds proximal regularization to reduce local drift [4], SCAFFOLD uses control variates to correct client drift [5], and FedNova normalizes local updates to reduce objective inconsistency under heterogeneous local computation [6]. These methods motivate the data-proportion term in PriSparK, but they communicate dense updates and do not use the event sparsity represented by the firing rate ρ_k^r or the spike-stability score a_k^r .

Privacy in federated learning is commonly strengthened by secure aggregation and differential privacy. Secure aggregation prevents direct inspection of individual client updates during aggregation [20], while differential privacy clips client contributions and injects calibrated noise so that a single client has limited influence on the released trajectory [18,19,21]. PriSparK adopts client-level differential privacy, but differs from dense DP-FL because clipping and Gaussian perturbation are applied after the event-sparse projection Ω_k^r . This placement preserves the formal privacy accounting while reducing utility loss from perturbing inactive or weakly informative coordinates.

Communication-efficient federated and distributed optimization has been studied through quantization, sparsification, and error feedback. QSGD reduces message precision with stochastic quantization [22], Deep Gradient Compression shows that gradient sparsification with correction can substantially reduce bandwidth [16], and sparsified SGD with memory provides a convergence basis for biased compression with residual compen-

sation [17]. PriSparK builds on this peer-reviewed compression literature but changes the selection criterion: a coordinate is transmitted only when its local update is coupled with presynaptic spike activity, postsynaptic surrogate sensitivity, and temporally stable firing.

2.2. Spiking Neural Networks and Federated Spiking Learning

Spiking neural networks encode information through sparse binary events and temporal membrane dynamics, making them attractive for low-power edge intelligence. Early backpropagation-based SNN training and spatio-temporal backpropagation established practical optimization procedures for deep spiking models [7,8]. SLAYER introduced spike-layer error reassignment over time [9], and surrogate-gradient learning later provided a general framework for training non-differentiable spike functions with gradient-based optimization [10]. Learnable membrane time constants further improved temporal representation [11], and recent surveys consolidate the connection between deep learning practice and SNN training [12].

The hardware motivation for event-driven learning is also supported by peer-reviewed neuromorphic systems. TrueNorth demonstrated a large-scale low-power neurosynaptic processor [13], and Loihi provided an on-chip learning architecture for neuromorphic computation [14]. These systems motivate local computation that exploits sparse events, but they do not determine which model coordinates should be communicated in a federated training round. PriSparK therefore treats spike activity as an optimization signal for communication, privacy, and aggregation rather than only as an inference-time efficiency feature.

The intersection of federated learning and spiking neural networks remains less mature than either field separately. Directly applying dense FedAvg, FedProx, or FedNova to spiking models ignores temporal event sparsity, while directly applying regular top- q sparsification ignores whether selected coordinates are involved in spike generation. PriSparK fills this gap by constructing the sparse subspace Ω_k^r from spike-gradient saliency, correcting compression bias through membrane-aware residual feedback, and using spike-stability-aware aggregation weights ω_k^r .

2.3. Privacy and Security Risks in Spiking and Neuromorphic Learning

Sparse or binary internal representations do not eliminate privacy risk. Membership inference can determine whether a sample was used for training [23,24]; model inversion can reconstruct sensitive attributes or representative inputs [25]; and gradient leakage can recover private training examples from shared gradients [2,3]. Collaborative learning can also leak unintended features even when raw data remain local [26]. These peer-reviewed results provide the security foundation for PriSparK: a federated SNN update must be protected at the level of transmitted values, selected coordinate masks, and bounded metadata, because spike sparsity alone is not a formal privacy guarantee.

3. Preliminaries

This section introduces the notation and basic concepts used throughout the paper. We consider a cross-device federated learning system with K clients and one coordination server. The set of clients is denoted by $\mathcal{K} = \{1, 2, \dots, K\}$, and client k owns a private dataset $\mathcal{D}_k = \{(x_i, y_i)\}_{i=1}^{n_k}$ with $n_k = |\mathcal{D}_k|$. The total number of training samples is $n = \sum_{k=1}^K n_k$, and the data proportion of client k is $p_k = n_k/n$. The global communication round is indexed by r , while the discrete spike time inside a spiking neural network is indexed by τ . Model parameters are denoted by $\theta \in \mathbb{R}^d$, where d is the number of trainable parameters. At the beginning of round r , the server broadcasts the global model θ^r to a selected client

subset $\mathcal{S}^r \subseteq \mathcal{K}$. Each selected client performs local spiking training and returns an update Δ_k^r to the server. The goal is to learn a global model that minimizes the federated objective

$$\min_{\theta} F(\theta) = \sum_{k=1}^K p_k F_k(\theta), \quad F_k(\theta) = \frac{1}{n_k} \sum_{(x_i, y_i) \in \mathcal{D}_k} \ell(f_{\theta}(x_i), y_i),$$

where $F_k(\theta)$ is the local empirical risk of client k , $\ell(\cdot, \cdot)$ is the task loss, and f_{θ} is the spiking neural network parameterized by θ .

3.1. Federated Spiking Neural Learning

A spiking neural network processes information through temporally sparse binary spikes rather than continuous activations. For an L -layer spiking model, let $u_{\ell, \tau}$ and $s_{\ell, \tau}$ denote the membrane potential and binary spike tensor of layer ℓ at spike time τ , respectively. A common discrete leaky integrate-and-fire dynamics can be written as

$$u_{\ell, \tau} = \lambda u_{\ell, \tau-1} + W_{\ell} s_{\ell-1, \tau} - v_{\text{th}} s_{\ell, \tau-1}, \quad s_{\ell, \tau} = H(u_{\ell, \tau} - v_{\text{th}}),$$

where W_{ℓ} is the synaptic weight matrix or convolutional kernel of layer ℓ , $\lambda \in (0, 1)$ is the membrane leakage factor, v_{th} is the firing threshold, and $H(\cdot)$ is the Heaviside step function. The reset term $v_{\text{th}} s_{\ell, \tau-1}$ prevents the membrane potential from growing without bound after firing. For an input sample x , the network output is obtained by accumulating the last-layer spike activities or membrane potentials over T spike steps, where T denotes the temporal simulation length.

The non-differentiability of $H(\cdot)$ makes direct backpropagation infeasible. Following standard surrogate-gradient training, the forward pass keeps the binary firing function $s_{\ell, \tau} = H(u_{\ell, \tau} - v_{\text{th}})$, while the backward pass replaces its derivative with a smooth or piecewise-smooth surrogate $\phi_{\alpha}(u_{\ell, \tau} - v_{\text{th}})$, where α controls the surrogate width or slope. Thus, local spiking optimization on client k updates θ by approximating $\nabla F_k(\theta)$ through backpropagation through time over $\tau = 1, \dots, T$. After E local epochs or a fixed number of local steps, client k obtains $\theta_k^{r, E}$ and forms the model update $\Delta_k^r = \theta_k^{r, E} - \theta^r$. In conventional federated averaging, the server aggregates client updates as $\theta^{r+1} = \theta^r + \sum_{k \in \mathcal{S}^r} \bar{p}_k \Delta_k^r$, where $\bar{p}_k = n_k / \sum_{j \in \mathcal{S}^r} n_j$ is the normalized data weight among selected clients.

In this paper, the central observation is that spiking training contains two different kinds of sparsity. The first is neural event sparsity, because most entries of $s_{\ell, \tau}$ are zero when the firing rate is low. The second is update sparsity, because only part of Δ_k^r is strongly coupled with informative spike events. PriSpark is designed around these two sparsity sources. We use $\rho_{\ell, \tau}^k = \|s_{\ell, \tau}^k\|_0 / |s_{\ell, \tau}^k|$ to denote the spike firing rate of layer ℓ at time τ on client k , where $\|\cdot\|_0$ counts nonzero entries and $|\cdot|$ denotes the number of tensor elements. The average firing rate of client k in round r is denoted by ρ_k^r , computed over all layers, spike times, and local mini-batches. Later sections use ρ_k^r to allocate communication budgets and to stabilize aggregation under heterogeneous spike activities.

3.2. Privacy Model

Federated learning avoids direct transmission of raw data, but the shared update Δ_k^r can still reveal sensitive information about $\mathcal{D}_k$. We adopt client-level differential privacy, where the presence or absence of one client's full local dataset is protected from an adversary observing the released update or the aggregated model trajectory. A randomized mechanism $\mathcal{M}$ satisfies (ϵ, δ) -differential privacy if, for any two neighboring federated datasets $\mathcal{D}$ and $\mathcal{D}'$ differing in one client and for any measurable output set $\mathcal{O}$, it holds that $\Pr[\mathcal{M}(\mathcal{D}) \in \mathcal{O}] \leq e^{\epsilon} \Pr[\mathcal{M}(\mathcal{D}') \in \mathcal{O}] + \delta$. A smaller ϵ indicates stronger privacy, and δ is usually chosen to be much smaller than $1/K$.

The server threat model is honest-but-curious. The server follows the sampling, aggregation, and accounting protocol, but it may try to infer client participation or data information from the received sparse messages, bounded metadata, and global model trajectory. Client-to-server channels are assumed to be authenticated, and the adversary does not arbitrarily change client messages, forge clients, or adaptively deviate from the protocol. Fully malicious servers, Byzantine clients, collusion between the server and a subset of clients, and side-channel leakage from local hardware are outside the formal guarantee; these stronger adversaries require additional mechanisms such as secure aggregation, verifiable execution, or Byzantine-robust aggregation.

A standard way to achieve differential privacy is to clip each client update and add calibrated Gaussian noise. Given a clipping threshold $C > 0$, the clipped update is $\text{clip}(\Delta_k^r, C) = \Delta_k^r \min\{1, C/\|\Delta_k^r\|_2\}$. Gaussian perturbation then releases $\tilde{\Delta}_k^r = \text{clip}(\Delta_k^r, C) + \xi_k^r$, where $\xi_k^r \sim \mathcal{N}(0, \sigma^2 C^2 I)$ and σ is the noise multiplier. However, adding noise in the full d -dimensional parameter space can severely damage the accuracy of spiking neural networks because surrogate gradients are already noisy and temporally accumulated. PriSpark therefore applies privacy perturbation after event-sparse projection. Specifically, each client constructs a binary sparsification mask $m_k^r \in \{0, 1\}^d$ and releases a compressed private update supported on $\Omega_k^r = \{j : m_{k,j}^r = 1\}$. The number of transmitted coordinates is $q_k^r = \|m_k^r\|_0$, and usually $q_k^r \ll d$.

The privacy analysis in the later methodology section treats event-sparse projection as a data-dependent preprocessing step followed by clipping and Gaussian perturbation in the selected subspace. To avoid uncontrolled privacy leakage through the mask itself, the mask is generated from bounded spike-gradient saliency and is transmitted either through a fixed budget rule or through a privatized top- q_k^r selection rule. The released client update is denoted by $\hat{\Delta}_k^r$, where the hat consistently indicates the final transmitted update after sparsification, clipping, noise injection, and quantization. The server only observes $\hat{\Delta}_k^r$ and never observes raw samples, raw spike trains, unclipped gradients, or local membrane traces.

3.3. Communication and Heterogeneity Measures

Communication efficiency is measured by the uplink cost from clients to the server, because edge federated systems are usually constrained by client bandwidth and energy. A dense 32-bit update requires approximately $32d$ transmitted bits per selected client per round. For sparse transmission, client k sends the selected coordinates and their quantized values. If each selected coordinate index uses b_{idx} bits and each quantized value uses b_{val} bits, the uplink cost of client k in round r is

$$B_k^r = q_k^r(b_{\text{idx}} + b_{\text{val}}) + B_{\text{meta}},$$

where B_{meta} is a metadata cost containing the random seed, layer-wise budget descriptor, clipping norm indicator, and privacy accounting information. In the implementation used for Table 1 and Figure 1, each selected coordinate uses $b_{\text{idx}} = \lceil \log_2 d \rceil$ index bits and each value uses $b_{\text{val}} = 8$ quantization bits. The metadata is encoded as a 32-bit random seed, a 32-bit sparse budget q_k^r , 16-bit codes for C_k^r and M_k^r , 8-bit codes for ρ_k^r and a_k^r , a 16-bit privacy-accounting code, and $16L$ bits for the layer-wise budget descriptor. Thus

$$B_{\text{meta}} = 32 + 32 + 16 + 16 + 8 + 8 + 16 + 16L = 128 + 16L \text{ bits.}$$

For the three-layer SNN backbones in Table 2, this gives $B_{\text{meta}} = 176$ bits per selected client per round. Even at the lower compression bound $\gamma_{\min} = 0.01$, the metadata fraction is at most $176/[0.01d(8 + \lceil \log_2 d \rceil)]$, so for $d \geq 10^5$ it remains below 0.8% of the sparse

index-value payload. Since B_{meta} is much smaller than $q_k^r(b_{\text{idx}} + b_{\text{val}})$ for practical neural networks, the main communication reduction comes from keeping q_k^r/d small. We denote the compression ratio by $\gamma_k^r = q_k^r/d$, where a smaller γ_k^r means fewer transmitted update coordinates.

Table 1. Main comparison under Dirichlet non-IID partition with $\alpha_D = 0.3$. Accuracy is reported as mean test accuracy in percent, and normalized uplink is measured relative to dense 32-bit FedAvg-SNN. Private methods use $(\epsilon_{\text{tot}}, \delta) = (4.0, 10^{-5})$.

Method	Privacy	N-MNIST	DVS128	CIFAR-10	Fashion	Uplink
FedAvg-ANN	None	99.18	94.21	76.54	92.38	$1.00 \times$
FedAvg-SNN	None	99.05	94.62	72.83	91.71	$1.00 \times$
FedProx-SNN	None	99.07	95.05	73.61	92.02	$1.00 \times$
FedNova-SNN	None	99.10	95.18	74.03	92.10	$1.00 \times$
Sparse FedSNN	None	98.62	93.18	70.74	90.55	$0.079 \times$
DP-FedAvg-SNN	Yes	97.83	91.47	68.21	89.12	$1.00 \times$
DP-Sparse FedSNN	Yes	96.94	90.22	66.43	88.35	$0.079 \times$
PriSparK	Yes	98.91	94.08	72.46	91.35	0.082 $\times$

Table 2. Default federated training and PriSparK hyperparameters. The same client sampling and local training protocol are used for all baselines to ensure a controlled comparison.

Item	Setting
Client population and participation	$K = 100$, $ \mathcal{S}^r = 10$ clients per round
Data partition	Dirichlet label skew with $\alpha_D = 0.3$ for main results; $\alpha_D \in \{0.1, 1.0\}$ for heterogeneity analysis
Local training	$E = 2$ local epochs, Adam optimizer, mini-batch size 64 for N-MNIST, CIFAR-10, and Fashion-MNIST, mini-batch size 16 for DVS128 Gesture
Learning rates	Local learning rate 10^{-3} for Fashion-MNIST and N-MNIST, 5×10^{-4} for CIFAR-10 and DVS128 Gesture, global step size $\eta_g = 1.0$
Spiking dynamics	Leaky integrate-and-fire neurons with $\lambda = 0.5$, $v_{\text{th}} = 1.0$, triangular surrogate derivative with $\alpha = 2.0$
Temporal length	$T = 16$ for CIFAR-10 and Fashion-MNIST, $T = 20$ for N-MNIST, $T = 32$ for DVS128 Gesture
Communication control	Base compression ratio $\gamma_0 = 0.05$, bounds $\gamma_{\min} = 0.01$ and $\gamma_{\max} = 0.10$, stochastic quantization with $b = 8$ bits
Privacy control	Client-level differential privacy with $\delta = 10^{-5}$, clipping threshold $C = 1.0$, noise multiplier selected according to the target ϵ_{tot} , mask allocation coefficient $\lambda_m = 0.15$
Adaptive compression and residuals	$\beta_\rho = 0.5$, $\beta_a = 0.5$, $\mu_0 = 0.9$, $\kappa_c = 0.2$, $\tau_\rho = \tau_a = 0.1$, $v_{\max} = 4$
Aggregation reliability	$\lambda_a = 1.0$, $\lambda_h = 0.5$, $\lambda_n = 0.5$, $a_{\min} = 0.2$, $a_{\max} = 1.0$, $h_{\min} = 0.25$, $h_{\max} = 4.0$, $\tau_M = 0.2$, $\epsilon = 10^{-12}$
Evaluation metrics	Top-1 test accuracy, normalized uplink communication, mean firing rate, privacy budget, and convergence rounds

Non-independent and identically distributed data introduce client drift, which becomes more pronounced in spiking models because local spike rates and temporal firing patterns can vary substantially across clients. We use $\bar{\Delta}^r = \sum_{k \in \mathcal{S}^r} \bar{p}_k \Delta_k^r$ to denote the uncompressed average update in round r when it is used for analysis, and define the relative update drift of client k as $d_k^r = \|\Delta_k^r - \bar{\Delta}^r\|_2 / (\|\bar{\Delta}^r\|_2 + \zeta)$, where $\zeta > 0$ is a small constant for numerical stability. A larger d_k^r indicates stronger deviation from the selected-client mean update. Because the server cannot access Δ_k^r after privacy protection in the actual protocol, PriSparK estimates heterogeneity through observable privatized quantities and through local statistics reported under bounded sensitivity.

Besides update drift, we use spike-activity stability to describe whether a client's temporal firing behavior is reliable across local training. Let $\rho_k^{r,e}$ be the average firing rate of client k at local epoch e in round r . The spike-activity variation is $v_k^r = \text{Var}_e(\rho_k^{r,e})$, and the corresponding stability score is $a_k^r = 1/(1 + v_k^r)$. A larger a_k^r means that local spike activities are temporally stable and less likely to produce unstable surrogate gradients. The

methodology section combines q_k^r , γ_k^r , and privacy-calibrated update norms to construct an adaptive aggregation weight. This design allows the server to reduce the influence of unstable or highly drifted client updates without inspecting private data, while preserving the event-driven communication advantage of spiking neural learning.

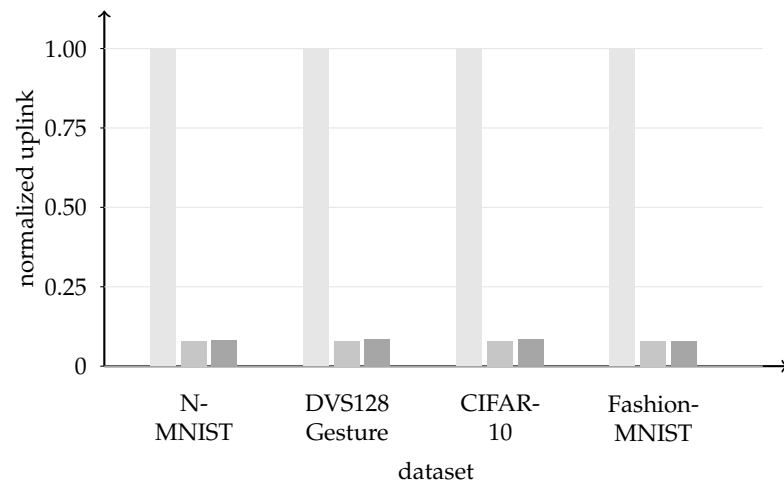


Figure 1. Per-round uplink reduction relative to dense 32-bit FedAvg-SNN. In each dataset group, the left bar is dense FedAvg-SNN, the middle bar is Sparse FedSNN, and the right bar is PriSparK; PriSparK includes sparse indices, quantized values, and bounded privacy metadata.

4. Methodology

4.1. Overview of PriSparK

PriSparK is designed for privacy-preserving and communication-efficient federated spiking neural learning under heterogeneous client data. In round r , the server broadcasts the current global spiking model θ^r to a selected client subset $\mathcal{S}^r$. Each client $k \in \mathcal{S}^r$ performs local surrogate-gradient training on its private dataset $\mathcal{D}_k$, obtains a local update Δ_k^r , transforms this update into an event-sparse representation guided by spike-gradient saliency, applies clipping and Gaussian perturbation in the selected low-dimensional subspace, and finally transmits the protected sparse update $\hat{\Delta}_k^r$ to the server. The server aggregates the received updates through a heterogeneity-adaptive rule and produces θ^{r+1} . Unlike conventional federated averaging, which communicates dense parameter updates and treats all selected client updates mainly according to data size, PriSparK couples the communication budget, privacy perturbation, and aggregation weight with spiking-specific activity statistics.

The key idea is that not all coordinates of a spiking model contribute equally to federated optimization. In a spiking neural network, an update coordinate is useful only when it is associated with informative membrane evolution and nontrivial spike activity. If a layer rarely fires or produces unstable spike patterns, transmitting a dense gradient from that layer wastes bandwidth and may amplify noisy surrogate gradients. PriSparK therefore defines an event-sparse update subspace Ω_k^r for each client and each round. The subspace is selected according to spike-gradient saliency rather than gradient magnitude alone. For parameter coordinate j , the saliency score is denoted by $g_{k,j}^r$, and the selected mask is $m_{k,j}^r = 1$ if $j \in \Omega_k^r$ and $m_{k,j}^r = 0$ otherwise. The sparse projection of a local update is $m_k^r \odot \Delta_k^r$, where $\odot$ denotes element-wise multiplication. The size of the selected subspace is $q_k^r = \|m_k^r\|_0$, and the compression ratio is $\gamma_k^r = q_k^r/d$.

PriSparK contains three coupled mechanisms. The first mechanism is event-sparse spike-gradient compression. Instead of assigning a fixed sparsity ratio to all clients and layers, client k allocates its communication budget according to its layer-wise spike activity $\rho_\ell^{k,r}$ and temporal stability. Here $\rho_\ell^{k,r}$ is the average firing rate of layer ℓ over local mini-

batches and spike steps in round r . Coordinates connected to stable and informative spike events receive larger selection probability, while coordinates associated with silent or highly unstable events are suppressed. This design reduces uplink traffic while preserving the parameters that most strongly affect event-driven inference.

The second mechanism is privacy-calibrated sparse perturbation. Directly applying differential privacy to a dense update $\Delta_k^r \in \mathbb{R}^d$ requires injecting noise into all d coordinates, which can noticeably degrade spiking learning because surrogate gradients are already affected by temporal approximation. PriSparK instead clips and perturbs the projected update inside Ω_k^r . The transmitted update has the form $\hat{\Delta}_k^r = Q_b(m_k^r \odot \text{clip}(\Delta_k^r, C_k^r) + \xi_k^r)$ on selected coordinates, where $Q_b(\cdot)$ is a b -bit quantizer, C_k^r is the adaptive clipping threshold, and ξ_k^r is Gaussian noise with variance calibrated by the privacy budget. The unselected coordinates are not transmitted. Because the perturbation is applied after event-sparse projection, the same privacy budget can be spent on fewer coordinates, leading to a more favorable accuracy–privacy trade-off.

The third mechanism is heterogeneity-adaptive aggregation. Non-independent and identically distributed local data cause client updates to drift away from the global descent direction, and this issue is more severe for spiking models because different clients may induce different firing patterns even under the same global parameters. PriSparK assigns an aggregation reliability score ω_k^r to each received update by jointly considering the selected client's data proportion $\bar{p}_k$, its spike-activity stability a_k^r , its normalized sparse update magnitude, and its privacy noise level. The server update is written as

$$\theta^{r+1} = \theta^r + \eta_g \sum_{k \in S^r} \omega_k^r \hat{\Delta}_k^r, \quad \sum_{k \in S^r} \omega_k^r = 1,$$

where η_g is the global aggregation step size. The weight ω_k^r is not a replacement for data-size weighting; rather, it is a corrected aggregation coefficient that starts from $\bar{p}_k$ and then adjusts the contribution of each client according to spiking stability and update reliability.

Figure 2 gives a graphical view of this coupling. Event-sparse compression determines the released subspace Ω_k^r , the subspace fixes where privacy noise is injected, and the reliability statistics determine how strongly each protected sparse update influences aggregation. The same three links appear in the optimization bound of Theorem 1. The proof of Theorem 1 is given in Appendix A.

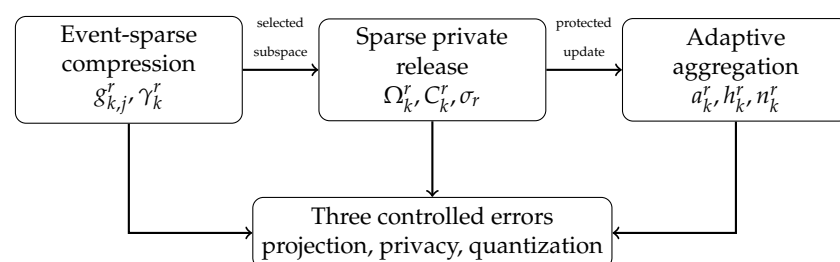


Figure 2. Coupled design of PriSparK. Compression chooses the event-relevant subspace, privacy perturbation is calibrated inside that subspace, and aggregation weights reduce the influence of unstable or noise-dominated sparse updates.

The overall optimization flow is summarized in Algorithm 1. The algorithm is intentionally organized so that the server never receives raw spikes, membrane potentials, local gradients, or unprotected dense updates. The client-side procedure only releases $\hat{\Delta}_k^r$ and a small bounded metadata vector containing the sparse budget descriptor, privacy-accounting state, and spike-stability statistic. The metadata is either insensitive by construction or clipped before transmission. This separation is important because the spike train

itself may encode sensitive sample-level information, especially on event-camera datasets where temporal firing patterns are closely related to the input motion.

Algorithm 1 PriSparK: Event-Sparse Private Federated Spiking Learning

Require: Initial global model θ^0 , clients $\mathcal{K}$, total rounds R , local epochs E , spike steps T , base clipping threshold C , noise multiplier σ , base compression ratio γ_0 , global step size η_g

Ensure: Final global spiking model θ^R

- 1: **for** $r = 0, 1, \dots, R - 1$ **do**
- 2: Server samples client subset $\mathcal{S}^r \subseteq \mathcal{K}$ and broadcasts θ^r
- 3: **for** each client $k \in \mathcal{S}^r$ in parallel **do**
- 4: Initialize local model $\theta_k^{r,0} \leftarrow \theta^r$
- 5: Train the local spiking model for E epochs and T spike steps using surrogate gradients
- 6: Form the dense local update $\Delta_k^r = \theta_k^{r,E} - \theta^r$
- 7: Estimate spike activity ρ_k^r , spike stability a_k^r , and spike-gradient saliency scores $\{g_{k,j}^r\}_{j=1}^d$
- 8: Construct event-sparse mask m_k^r with budget $q_k^r = \|m_k^r\|_0$
- 9: Clip, perturb, quantize, and transmit the private sparse update $\hat{\Delta}_k^r$
- 10: **end for**
- 11: Server computes aggregation weights $\{\omega_k^r : k \in \mathcal{S}^r\}$ from bounded reliability statistics
- 12: Server updates $\theta^{r+1} = \theta^r + \eta_g \sum_{k \in \mathcal{S}^r} \omega_k^r \hat{\Delta}_k^r$
- 13: **end for**
- 14: **return** θ^R

The novelty of PriSparK lies in the fact that compression, privacy, and aggregation are not treated as independent modules. Event sparsity determines where communication is spent, the selected subspace determines where privacy noise is injected, and the stability of spike activity determines how much the server trusts a protected sparse update. This coupling is especially suitable for federated spiking neural networks, because the temporal event structure provides an additional signal that is absent in conventional artificial neural networks. The following subsections describe the event-sparse compression rule, the privacy-preserving error-feedback update, and the heterogeneity-adaptive aggregation mechanism in detail.

4.2. Event-Sparse Adaptive Spike-Gradient Compression

The first component of PriSparK is an event-sparse compression rule that selects update coordinates according to their contribution to spike-driven computation. Conventional top- q sparsification keeps the largest entries of Δ_k^r by magnitude, but this criterion is not fully aligned with spiking neural learning. A large surrogate-gradient entry may be produced by unstable membrane dynamics or by noisy temporal credit assignment, while a moderate update associated with consistent spike events may be more useful for global convergence. PriSparK therefore constructs the sparse mask m_k^r from a spike-gradient saliency score that combines update magnitude, presynaptic event activity, postsynaptic surrogate sensitivity, and temporal stability.

For a parameter coordinate j belonging to layer $\ell(j)$, PriSparK defines its local saliency on client k in round r as

$$g_{k,j}^r = |\Delta_{k,j}^r| \cdot \sqrt{\epsilon + \chi_{k,j}^r} \cdot \frac{1}{1 + v_{\ell(j)}^{k,r}},$$

where $\epsilon > 0$ is a small numerical constant, $\chi_{k,j}^r$ is the event-gradient coupling score, and $v_{\ell}^{k,r}$ is the layer-wise temporal spike variation. The first term $|\Delta_{k,j}^r|$ measures the direct local optimization signal. The second term $\sqrt{\epsilon + \chi_{k,j}^r}$ increases the priority of parameters that are repeatedly involved in informative spike propagation. The third term down-weights coordinates in layers with unstable temporal firing patterns, because high spike variation often indicates client-specific drift or noisy surrogate gradients under non-independent and identically distributed data.

The event-gradient coupling score is computed from local spike trains and surrogate derivatives without transmitting them to the server. For a synaptic parameter connecting a presynaptic component a to a postsynaptic component b in layer ℓ , the score is

$$\chi_{k,j}^r = \frac{1}{|\mathcal{B}_k^r|T} \sum_{(x,y) \in \mathcal{B}_k^r} \sum_{\tau=1}^T |s_{\ell-1,a,\tau}^k(x)| \cdot |\phi_a(u_{\ell,b,\tau}^k(x) - v_{th})|,$$

where $\mathcal{B}_k^r$ denotes the collection of mini-batches used by client k in round r , $s_{\ell-1,a,\tau}^k(x)$ is the presynaptic spike event, and $\phi_a(\cdot)$ is the surrogate derivative used in local backpropagation. This definition assigns high saliency to a parameter only when the presynaptic neuron fires and the postsynaptic membrane potential is near the firing threshold, because in that case the parameter has a strong influence on future spike generation. For convolutional layers, $\chi_{k,j}^r$ is averaged over spatial positions sharing the same kernel coordinate. For batch-normalization-free spiking architectures, which are used in our experiments for stable temporal dynamics, the same definition applies directly to convolutional kernels and linear readout weights.

The saliency computation is implemented as an online accumulator during BPTT rather than by storing all spike trains and surrogate derivatives. The required extra memory is one FP32 accumulator for each trainable coordinate, $32d$ bits, plus $O(LT)$ scalar firing-rate accumulators; the local residual buffer e_k^r already required by error feedback adds another $32d$ bits but is not part of the event-gradient score. The additional computation is a multiply-accumulate using the presynaptic spike indicator and the postsynaptic surrogate derivative that are already produced in the backward pass. Thus the asymptotic client-side training cost remains that of local BPTT, with an added streaming accumulation pass over parameter uses. For the temporal lengths in Table 2, no per-sample or per-time-step saliency tensor is stored after the mini-batch has been processed.

The layer-wise temporal spike variation $v_{\ell}^{k,r}$ is computed from firing rates observed during local training with the spike-time index written consistently as τ . Let $\rho_{\ell,\tau}^{k,r}$ denote the average firing rate of layer ℓ at spike time τ on client k in round r . To avoid making short simulation windows appear artificially unstable, PriSpark uses a normalized and clipped variation

$$v_{\ell}^{k,r} = \text{clip} \left(\frac{T^{-1} \sum_{\tau=1}^T (\rho_{\ell,\tau}^{k,r} - \bar{\rho}_{\ell}^{k,r})^2}{\bar{\rho}_{\ell}^{k,r} (1 - \bar{\rho}_{\ell}^{k,r}) + \epsilon_v}, 0, v_{\max} \right), \quad \bar{\rho}_{\ell}^{k,r} = T^{-1} \sum_{\tau=1}^T \rho_{\ell,\tau}^{k,r}.$$

The denominator normalizes by the Bernoulli firing variance scale, and clipping prevents a small T from producing an excessively large penalty. In all experiments, T is fixed per dataset as reported in Table 2, so comparisons across methods use the same temporal resolution. This quantity is different from the client-level stability score a_k^r introduced in Section 3.3: $v_{\ell}^{k,r}$ controls coordinate selection inside the client, while a_k^r is later used by the server to adjust aggregation weights.

After computing $g_{k,j}^r$, client k determines an adaptive communication budget. The total number of selected coordinates is $q_k^r = \lfloor \gamma_k^r d \rfloor$, where the compression ratio γ_k^r is not fixed but depends on the client's average firing rate and stability:

$$\gamma_k^r = \text{clip} \left(\gamma_0 \left[1 + \beta_\rho \frac{\rho_k^r - \bar{\rho}^r}{\bar{\rho}^r + \epsilon} + \beta_a (a_k^r - \bar{a}^r) \right], \gamma_{\min}, \gamma_{\max} \right).$$

Here γ_0 is the base compression ratio, $\gamma_{\min}$ and $\gamma_{\max}$ are lower and upper bounds, β_ρ and β_a control the sensitivity to spike activity and stability, and $\bar{\rho}^r$ and $\bar{a}^r$ are moving averages estimated by the server from bounded client metadata in previous rounds. A client with more active and stable spike patterns receives a slightly larger communication budget, because its sparse update is likely to contain more globally useful event-driven information. A silent or unstable client is still allowed to participate, but its communication budget is reduced to avoid spending bandwidth on low-confidence coordinates.

The total budget q_k^r is further distributed across layers. If d_ℓ is the number of parameters in layer ℓ , the layer-wise budget is

$$q_{k,\ell}^r = \left\lfloor q_k^r \frac{d_\ell \sqrt{\epsilon + \bar{\rho}_\ell^{k,r} / (1 + \nu_\ell^{k,r})}}{\sum_{h=1}^L d_h \sqrt{\epsilon + \bar{\rho}_h^{k,r} / (1 + \nu_h^{k,r})}} \right\rfloor.$$

This allocation avoids the failure mode in which most selected coordinates come from the largest layers only. It also avoids blindly preserving silent early layers or noisy late layers. Within each layer, PriSparK selects the $q_{k,\ell}^r$ coordinates with the largest saliency values $g_{k,j}^r$. The resulting sparse mask satisfies $\|m_k^r\|_0 \leq q_k^r$, and the sparse projected update is $m_k^r \odot \Delta_k^r$.

To reduce the bias caused by repeated sparsification, PriSparK keeps a membrane-aware residual buffer $e_k^r \in \mathbb{R}^d$ on each client. Before saliency ranking, the client forms a corrected update $z_k^r = \Delta_k^r + \mu_k^r e_k^r$, where μ_k^r is a residual injection coefficient. Unlike standard error feedback, which uses a fixed residual coefficient, PriSparK adapts μ_k^r according to spike stability as $\mu_k^r = \mu_0 a_k^r$, where $\mu_0 \in [0, 1]$ is the maximum feedback strength. Stable clients can reuse a larger portion of their previous compression error, while unstable clients suppress residual accumulation to prevent outdated spike-gradient errors from dominating the current update. After selecting the mask, the local residual is updated by

$$e_k^{r+1} = z_k^r - m_k^r \odot z_k^r.$$

The residual buffer is kept locally and is never transmitted. Therefore, it does not increase communication cost and does not reveal additional information to the server.

Algorithm 2 summarizes the event-sparse compression procedure. The output of this procedure is the selected update $m_k^r \odot z_k^r$, which is then passed to the privacy-preserving perturbation module described in the next subsection. This order is important: compression is performed before clipping and noise injection, so the privacy noise is concentrated on the selected event-relevant subspace rather than spread across all model parameters.

4.3. Privacy-Calibrated Sparse Perturbation and Quantized Transmission

After event-sparse compression, client k obtains a corrected sparse update $m_k^r \odot z_k^r$, where $z_k^r = \Delta_k^r + \mu_k^r e_k^r$ includes the membrane-aware residual feedback defined in Section 4.2. This sparse update is still not safe to transmit directly, because both the selected coordinates and their values may reveal information about the private dataset $\mathcal{D}_k$. PriSparK therefore applies a privacy-calibrated release mechanism that protects the update values, controls the information exposed by the sparse mask, and reduces the number of

transmitted bits through stochastic quantization. The final transmitted object is denoted by $\hat{\Delta}_k^r$, and all subsequent server-side operations use only $\hat{\Delta}_k^r$ and bounded metadata.

Algorithm 2 Event-Sparse Adaptive Spike-Gradient Compression on Client k

Require: Local update Δ_k^r , residual buffer e_k^r , spike trains $\{s_{\ell,\tau}^k\}$, membrane potentials $\{u_{\ell,\tau}^k\}$, base compression ratio γ_0 , bounds $\gamma_{\min}$ and $\gamma_{\max}$

Ensure: Sparse corrected update $m_k^r \odot z_k^r$, mask m_k^r , updated residual e_k^{r+1}

- 1: Compute average firing rate ρ_k^r , client stability a_k^r , layer firing rates $\{\rho_{\ell}^{k,r}\}_{\ell=1}^L$, and layer variations $\{v_{\ell}^{k,r}\}_{\ell=1}^L$
 - 2: Set residual coefficient $\mu_k^r = \mu_0 a_k^r$ and corrected update $z_k^r = \Delta_k^r + \mu_k^r e_k^r$
 - 3: Compute adaptive compression ratio γ_k^r and total budget $q_k^r = \lfloor \gamma_k^r d \rfloor$
 - 4: **for** each layer $\ell = 1, \dots, L$ **do**
 - 5: Allocate layer budget $q_{k,\ell}^r$ from $\bar{\rho}_{\ell}^{k,r}$ and $v_{\ell}^{k,r}$
 - 6: Compute saliency $g_{k,j}^r$ for all coordinates j in layer ℓ
 - 7: Select the $q_{k,\ell}^r$ coordinates with the largest $g_{k,j}^r$ and set their mask values to one
 - 8: **end for**
 - 9: Construct sparse update $m_k^r \odot z_k^r$
 - 10: Update local residual $e_k^{r+1} = z_k^r - m_k^r \odot z_k^r$
 - 11: **return** $m_k^r \odot z_k^r, m_k^r, e_k^{r+1}$
-

The first step is sparse update clipping. Let $\Omega_k^r = \{j : m_{k,j}^r = 1\}$ be the selected coordinate set and let $P_{\Omega_k^r}(\cdot)$ be the projection operator that keeps coordinates in Ω_k^r and removes all other coordinates. PriSparK clips the selected corrected update as

$$\bar{z}_k^r = P_{\Omega_k^r}(z_k^r) \min \left\{ 1, \frac{C_k^r}{\|P_{\Omega_k^r}(z_k^r)\|_2 + \epsilon} \right\},$$

where C_k^r is the client-specific clipping threshold and $\epsilon > 0$ avoids numerical instability. The threshold is adapted from the base threshold C according to spike stability, namely $C_k^r = C(1 + \kappa_c(a_k^r - \bar{a}^r))$ clipped into $[C_{\min}, C_{\max}]$. Here κ_c is a sensitivity coefficient, a_k^r is the client-level spike-activity stability, and $\bar{a}^r$ is the server-maintained moving average of stability. This adaptive clipping rule allows stable clients to preserve slightly more update energy while preventing unstable clients from transmitting high-norm surrogate-gradient outliers.

The second step is privacy-aware sparse mask release. Although the numerical update values are protected by Gaussian noise, the coordinate set Ω_k^r itself may encode private information because it is selected from spike-gradient saliency. PriSparK handles this issue by replacing deterministic top- q_k^r selection with a bounded randomized ranking mechanism during privacy-preserving training. For each candidate coordinate j in layer ℓ , the raw saliency $g_{k,j}^r$ is clipped into $[0, G_{\ell}]$ and converted into a private ranking score $\tilde{g}_{k,j}^r = g_{k,j}^{r,\text{clip}} + \eta_{k,j}^r$, where $\eta_{k,j}^r$ is sampled from a Gumbel distribution calibrated by a mask privacy temperature τ_m . The client then selects the $q_{k,\ell}^r$ largest $\tilde{g}_{k,j}^r$ values in each layer. This randomized ranking can be interpreted as an exponential-mechanism-style sparse selection rule with bounded saliency sensitivity. In non-private communication-only variants, the deterministic saliency ranking in Algorithm 2 is recovered by setting τ_m close to zero, while the privacy-preserving variant keeps $\tau_m > 0$ and accounts for mask selection in the total privacy budget.

The third step is Gaussian perturbation in the selected subspace. Given the clipped sparse update $\bar{z}_k^r$, PriSparK samples noise only on the selected coordinates:

$$\tilde{z}_{k,j}^r = \begin{cases} \bar{z}_{k,j}^r + \xi_{k,j}^r, & j \in \Omega_k^r, \\ 0, & j \notin \Omega_k^r, \end{cases} \quad \xi_{k,j}^r \sim \mathcal{N}(0, \sigma_r^2 (C_k^r)^2).$$

The noise multiplier σ_r is chosen according to the target privacy budget and the sampling rate of clients in round r . When privacy amplification by subsampling is used, only clients in $\mathcal{S}^r$ participate in the release mechanism, and the privacy accountant accumulates the loss over communication rounds. PriSparK allocates the round-level privacy cost between sparse mask selection and update-value perturbation. If the per-round privacy budget is denoted by ϵ_r , then $\epsilon_r^{\text{mask}} = \lambda_m \epsilon_r$ is assigned to randomized mask selection and $\epsilon_r^{\text{val}} = (1 - \lambda_m) \epsilon_r$ is assigned to Gaussian perturbation, where $\lambda_m \in [0, 1)$ is the mask privacy allocation coefficient. This separation prevents the common mistake of protecting update values while leaving the selected sparse indices unaccounted for. The default value is $\lambda_m = 0.15$, so 85% of each round-level budget is reserved for value perturbation. This choice follows the rule that mask selection receives a small but nonzero budget because its saliency scores are clipped into $[0, G_\ell]$, while the released values have ℓ_2 sensitivity C_k^r and dominate utility. A practical non-adaptive guideline is $\lambda_m \in [0.10, 0.20]$; an adaptive guideline is $\lambda_m = \text{clip}(\sqrt{\gamma_k^r} / (1 + \sqrt{\gamma_k^r}), 0.10, 0.20)$, which assigns slightly more mask budget only when more coordinates are released. Values above 0.30 are avoided because they leave too little budget for Gaussian value perturbation under tight ϵ_{tot} .

Because the perturbation is applied after sparse projection, the released update has lower effective dimensionality than a dense differentially private update. The privacy guarantee is still determined by sensitivity and noise calibration rather than dimension alone, but the sparse design improves utility because the injected noise affects only event-relevant coordinates that will actually be transmitted and aggregated. For the same clipping threshold, a dense mechanism perturbs all d coordinates even though many of them are unimportant under event-sparse spiking computation. In contrast, PriSparK perturbs q_k^r selected coordinates, where $q_k^r \ll d$, and leaves nonselected coordinates absent from the message. This design improves the signal-to-noise ratio of the communicated update under a fixed uplink budget.

The fourth step is stochastic quantization. To further reduce communication, PriSparK quantizes the noisy sparse values before transmission. Let b be the number of quantization bits and let $M_k^r = \max_{j \in \Omega_k^r} |\tilde{z}_{k,j}^r| + \epsilon$. Each selected value is normalized into $[-1, 1]$, stochastically rounded to one of 2^b uniformly spaced levels, and then rescaled by M_k^r . The quantized value is written as $Q_b(\tilde{z}_{k,j}^r)$. Stochastic quantization is used instead of deterministic rounding because it reduces systematic bias in the aggregated sparse update. The transmitted update is therefore

$$\hat{\Delta}_{k,j}^r = \begin{cases} Q_b(\tilde{z}_{k,j}^r), & j \in \Omega_k^r, \\ 0, & j \notin \Omega_k^r. \end{cases}$$

Client k sends the selected coordinate indices, the quantized noisy values, and bounded metadata including q_k^r , C_k^r , M_k^r , and clipped spike-stability statistics. Since M_k^r may also depend on local data, PriSparK clips it into a pre-defined interval and optionally releases it through a low-precision codebook shared by all clients. This keeps the metadata stable and prevents it from becoming an unprotected side channel.

The residual buffer must also be updated carefully under privacy perturbation. If the client stored the difference between z_k^r and the noisy quantized message, the residual

would contain privacy noise and quantization artifacts, which would be repeatedly re-injected into later rounds. PriSparK instead stores only the deterministic compression residual from Section 4.2, namely $e_k^{r+1} = z_k^r - P_{\Omega_k^r}(z_k^r)$, before clipping, perturbation, and quantization. This choice preserves the purpose of error feedback, which is to compensate for omitted coordinates, while avoiding the accumulation of artificial privacy noise. The server never receives e_k^{r+1} , and the residual is overwritten locally as training proceeds. Omitting clipping noise and Gaussian privacy noise from the residual does not remove the zero-mean perturbation from the transmitted update; it only prevents artificial noise from being accumulated and re-injected in later rounds. The residual is bounded in practice by clipping z_k^r before release and by using $0 \leq \mu_k^r \leq \mu_0 \leq 1$. Under severe heterogeneity, unstable firing increases v_k^r , lowers a_k^r , and therefore reduces $\mu_k^r = \mu_0 a_k^r$, which suppresses long-term carry-over of outdated compression errors.

The complete privacy-calibrated sparse transmission procedure is given in Algorithm 3. The algorithm uses a candidate mask from event-sparse saliency, protects the selected indices through randomized ranking when privacy is required, clips the sparse corrected update, injects Gaussian noise, and quantizes the result. The released update $\hat{\Delta}_k^r$ is the only model update transmitted from client k to the server.

Algorithm 3 Privacy-Calibrated Sparse Release on Client k

Require: Corrected update z_k^r , saliency scores $\{g_{k,j}^r\}_{j=1}^d$, layer budgets $\{q_{k,\ell}^r\}_{\ell=1}^L$, stability score a_k^r , clipping threshold bounds $C_{\min}$ and $C_{\max}$, base threshold C , noise multiplier σ_r , quantization bits b

Ensure: Private quantized sparse update $\hat{\Delta}_k^r$

- 1: Compute adaptive clipping threshold $C_k^r = \text{clip}(C(1 + \kappa_c(a_k^r - \bar{a}^r)), C_{\min}, C_{\max})$
- 2: **for** each layer $\ell = 1, \dots, L$ **do**
- 3: Clip saliency scores into $[0, G_\ell]$
- 4: Sample randomized ranking scores $\tilde{g}_{k,j}^r = g_{k,j}^{r,\text{clip}} + \eta_{k,j}^r$
- 5: Select the $q_{k,\ell}^r$ largest randomized ranking scores and update Ω_k^r
- 6: **end for**
- 7: Compute clipped sparse update $\bar{z}_k^r = P_{\Omega_k^r}(z_k^r) \min\{1, C_k^r / (\|P_{\Omega_k^r}(z_k^r)\|_2 + \epsilon)\}$
- 8: Sample Gaussian noise $\xi_{k,j}^r \sim \mathcal{N}(0, \sigma_r^2 (C_k^r)^2)$ for each $j \in \Omega_k^r$
- 9: Form noisy sparse update $\tilde{z}_{k,j}^r = \bar{z}_{k,j}^r + \xi_{k,j}^r$ for $j \in \Omega_k^r$ and $\tilde{z}_{k,j}^r = 0$ otherwise
- 10: Quantize selected values with stochastic b -bit quantizer $Q_b(\cdot)$
- 11: Set $\hat{\Delta}_{k,j}^r = Q_b(\tilde{z}_{k,j}^r)$ for $j \in \Omega_k^r$ and $\hat{\Delta}_{k,j}^r = 0$ otherwise
- 12: **return** $\hat{\Delta}_k^r$

The privacy guarantee of PriSparK follows from the composition of the randomized sparse selection mechanism and the Gaussian update release mechanism. For each round, saliency clipping bounds the sensitivity of the mask-selection utility, sparse update clipping bounds the ℓ_2 sensitivity of the released values by C_k^r , and Gaussian perturbation provides calibrated privacy for the selected coordinates. Across rounds, the total privacy loss is accumulated with a moments accountant or Rényi differential privacy accountant. If the total number of rounds is R , the client sampling probability is $s = |\mathcal{S}^r|/K$, and the round-level parameters are fixed, the final privacy budget can be expressed as $(\epsilon_{\text{tot}}, \delta)$ after composing R subsampled mechanisms. In the experiments, δ is set according to the number of clients, and σ_r is adjusted to compare different privacy regimes under the same federated spiking architecture.

This privacy-calibrated sparse release is central to PriSparK because it prevents communication efficiency and privacy protection from working against each other. Compression alone may reveal sensitive coordinate patterns; differential privacy alone may destroy the weak but useful spike-gradient signals; quantization alone may introduce

biased aggregation errors. By combining randomized sparse selection, subspace clipping, Gaussian perturbation, and stochastic quantization in a single client-side mechanism, PriSparK produces updates that are compact, privacy-aware, and compatible with stable server aggregation.

4.4. Heterogeneity-Adaptive Aggregation and Optimization Behavior

After receiving the protected sparse updates $\{\hat{\Delta}_k^r : k \in \mathcal{S}^r\}$, the server must aggregate them without access to raw data, dense local gradients, membrane potentials, or spike trains. A direct weighted average using only the selected-client data proportions $\bar{p}_k = n_k / \sum_{j \in \mathcal{S}^r} n_j$ is simple, but it ignores two sources of instability that are particularly important in federated spiking neural learning. First, a client may produce unstable spike activities across local epochs, leading to noisy surrogate-gradient updates even when its local loss decreases. Second, privacy perturbation and sparse quantization may cause the transmitted update $\hat{\Delta}_k^r$ to deviate from the underlying local update direction. PriSparK addresses these issues through a heterogeneity-adaptive aggregation rule that corrects data-size weighting using bounded reliability statistics.

For each participating client k , the server constructs a reliability score from three observable or bounded quantities: the spike-activity stability a_k^r , the sparse update norm after protection, and the effective privacy-noise scale. The stability score $a_k^r = 1 / (1 + v_k^r)$ follows the definition in Section 3.3, where v_k^r is computed from the variation of local firing rates across local epochs. Before transmission, a_k^r is clipped into $[a_{\min}, a_{\max}]$ to bound its sensitivity. The protected sparse update norm is normalized as $h_k^r = \|\hat{\Delta}_k^r\|_2 / (\text{median}_{j \in \mathcal{S}^r} \|\hat{\Delta}_j^r\|_2 + \epsilon)$, where the median improves robustness to outlying client updates. The privacy-noise factor is written as $n_k^r = \sigma_r C_k^r / (\|\hat{\Delta}_k^r\|_2 + \epsilon)$, where σ_r is the noise multiplier and C_k^r is the adaptive clipping threshold. A larger n_k^r means that the transmitted update has a lower signal-to-noise ratio. The median is used instead of the mean because sparse protected updates can contain outlying norms caused by label skew, clipping, or privacy noise. With $m = |\mathcal{S}^r|$ participating clients, the median has a 50% breakdown point, whereas the mean can be shifted by one extreme client. When m is small, its variance can increase; the implementation therefore clips h_k^r into $[h_{\min}, h_{\max}]$ and may smooth the denominator as $M_{\text{ema}}^r = (1 - \tau_M) M_{\text{ema}}^{r-1} + \tau_M \text{median}_{j \in \mathcal{S}^r} \|\hat{\Delta}_j^r\|_2$ with $\tau_M = 0.2$. The default experiments use $m = 10$, and the low-participation stress setting in Table 3 uses the same clipping and smoothing rule.

Table 3. Stress-test results under stronger edge constraints. The first row uses a lower participation fraction with $|\mathcal{S}^r| = 5$ out of $K = 100$ clients. The second row uses stronger label skew with $\alpha_D = 0.1$. All private runs use $(\epsilon_{\text{tot}}, \delta) = (4.0, 10^{-5})$.

Stress Setting	N-MNIST	DVS128	CIFAR-10	Fashion	Average
$ \mathcal{S}^r = 5, \alpha_D = 0.3$	98.57	93.24	71.21	90.78	88.45
$ \mathcal{S}^r = 10, \alpha_D = 0.1$	98.18	92.67	69.84	89.92	87.65

The unnormalized aggregation reliability is defined as

$$\psi_k^r = \bar{p}_k \cdot (a_k^r)^{\lambda_a} \cdot \exp(-\lambda_h |h_k^r - 1|) \cdot \frac{1}{1 + \lambda_n n_k^r},$$

where λ_a , λ_h , and λ_n are nonnegative coefficients controlling the influence of spike stability, robust norm consistency, and privacy-noise awareness. The final aggregation weight is obtained by normalization,

$$\omega_k^r = \frac{\psi_k^r}{\sum_{j \in \mathcal{S}^r} \psi_j^r + \epsilon}.$$

This design preserves the role of data-size weighting through $\bar{p}_k$, but it avoids over-trusting a large client when its spike activity is unstable or when its transmitted update is dominated by privacy noise. The factor $\exp(-\lambda_h|h_k^r - 1|)$ penalizes updates with unusually small or large protected norms. This is useful because extreme sparse-update norms often indicate either aggressive clipping, excessive quantization distortion, or client drift caused by non-independent and identically distributed local data.

The server update is then computed as

$$\theta^{r+1} = \theta^r + \eta_g \sum_{k \in S^r} \omega_k^r \hat{\Delta}_k^r$$

where η_g is the global step size. Since $\hat{\Delta}_k^r$ is sparse, the server implements aggregation with an index-value accumulator rather than expanding all updates into dense vectors. If different clients select different coordinate sets, the accumulator sums only the coordinates that appear in at least one received update. Missing coordinates are treated as zeros. This implementation keeps server-side computation proportional to the number of received sparse coordinates, namely $\sum_{k \in S^r} q_k^r$, rather than $|S^r|d$.

PriSparK also maintains moving averages of global spike statistics to stabilize future rounds. After aggregation, the server updates $\bar{\rho}^{r+1} = (1 - \tau_\rho)\bar{\rho}^r + \tau_\rho|S^r|^{-1} \sum_{k \in S^r} \rho_k^r$ and $\bar{a}^{r+1} = (1 - \tau_a)\bar{a}^r + \tau_a|S^r|^{-1} \sum_{k \in S^r} a_k^r$, where τ_ρ and τ_a are smoothing coefficients. These moving averages are used in Section 4.2 to compute the adaptive compression ratio γ_k^r and in Section 4.3 to compute the adaptive clipping threshold C_k^r . Because only clipped scalar statistics are transmitted, these moving averages add negligible communication overhead compared with model updates.

The complete server-side aggregation procedure is summarized in Algorithm 4. The algorithm receives only privacy-protected sparse updates and bounded client statistics. It first estimates the reliability of each client update, then normalizes the reliability scores into aggregation weights, and finally updates the global spiking model. The same procedure is used for all datasets and architectures in the experiments.

Algorithm 4 Heterogeneity-Adaptive Server Aggregation

Require: Global model θ^r , received updates $\{\hat{\Delta}_k^r : k \in S^r\}$, data proportions $\{\bar{p}_k : k \in S^r\}$, clipped statistics $\{a_k^r, \rho_k^r, C_k^r : k \in S^r\}$, noise multiplier σ_r , global step size η_g

Ensure: Updated global model θ^{r+1}

- 1: Compute protected update norms $\{\|\hat{\Delta}_k^r\|_2 : k \in S^r\}$
 - 2: Compute median norm $M^r = \text{median}_{j \in S^r} \|\hat{\Delta}_j^r\|_2$
 - 3: **for** each client $k \in S^r$ **do**
 - 4: Set $h_k^r = \|\hat{\Delta}_k^r\|_2 / (M^r + \epsilon)$
 - 5: Set $n_k^r = \sigma_r C_k^r / (\|\hat{\Delta}_k^r\|_2 + \epsilon)$
 - 6: Compute $\psi_k^r = \bar{p}_k (a_k^r)^{\lambda_a} \exp(-\lambda_h |h_k^r - 1|) / (1 + \lambda_n n_k^r)$
 - 7: **end for**
 - 8: Normalize $\omega_k^r = \psi_k^r / (\sum_{j \in S^r} \psi_j^r + \epsilon)$ for all $k \in S^r$
 - 9: Aggregate sparse updates with $\theta^{r+1} = \theta^r + \eta_g \sum_{k \in S^r} \omega_k^r \hat{\Delta}_k^r$
 - 10: Update moving averages $\bar{\rho}^{r+1}$ and $\bar{a}^{r+1}$ using clipped client statistics
 - 11: **return** θ^{r+1}
-

The following result describes the optimization behavior of PriSparK under standard smoothness and bounded-variance assumptions. It is not intended to hide the effects of privacy noise and compression; instead, it makes explicit how the sparse ratio, perturbation scale, quantization error, and heterogeneity correction enter the descent bound.

Theorem 1 (One-round descent of PriSparK). Assume that the global objective $F(\theta)$ is L_F -smooth, each local stochastic surrogate gradient has bounded variance σ_g^2 , and the local update norm is bounded after clipping by C_k^r . Let the sparse projection error satisfy $\mathbb{E}\|z_k^r - P_{\Omega_k^r}(z_k^r)\|_2^2 \leq \eta_{s,k}^r \|z_k^r\|_2^2$, where $\eta_{s,k}^r$ decreases as the selected event-sparse budget q_k^r increases. Let the stochastic quantization error satisfy $\mathbb{E}\|Q_b(\tilde{z}_k^r) - \tilde{z}_k^r\|_2^2 \leq \eta_q(b) \|\tilde{z}_k^r\|_2^2$, where $\eta_q(b)$ decreases with the number of quantization bits b . Then, for a sufficiently small global step size η_g , the expected one-round objective change satisfies

$$\begin{aligned} \mathbb{E}[F(\theta^{r+1})] \leq & F(\theta^r) - \eta_g \left\langle \nabla F(\theta^r), \sum_{k \in \mathcal{S}^r} \omega_k^r \mathbb{E}[P_{\Omega_k^r}(z_k^r)] \right\rangle \\ & + \frac{L_F \eta_g^2}{2} \mathbb{E} \left\| \sum_{k \in \mathcal{S}^r} \omega_k^r \hat{\Delta}_k^r \right\|_2^2 + \mathcal{O} \left(\sum_{k \in \mathcal{S}^r} (\omega_k^r)^2 \left[\eta_{s,k}^r \|z_k^r\|_2^2 + q_k^r \sigma_r^2 (C_k^r)^2 + \eta_q(b) \|\tilde{z}_k^r\|_2^2 \right] \right). \end{aligned}$$

Theorem 1, together with the mechanism diagram in Figure 2, shows that PriSparK improves the federated spiking trade-off by controlling three error terms simultaneously. The sparse projection error $\eta_{s,k}^r \|z_k^r\|_2^2$ is reduced by selecting coordinates with high spike-gradient saliency instead of using random or magnitude-only sparsification. The privacy perturbation term $q_k^r \sigma_r^2 (C_k^r)^2$ grows with the number of selected coordinates and the noise multiplier, which explains why injecting noise in the event-sparse subspace is more useful than perturbing all d coordinates. The quantization term $\eta_q(b) \|\tilde{z}_k^r\|_2^2$ is controlled by stochastic rounding and by using enough quantization bits for noisy sparse values. The aggregation coefficients ω_k^r further reduce the influence of unreliable clients because all three errors are weighted by $(\omega_k^r)^2$ in the high-order term.

A proof sketch is as follows. By L_F -smoothness, $F(\theta^{r+1})$ is upper-bounded by $F(\theta^r) + \langle \nabla F(\theta^r), \theta^{r+1} - \theta^r \rangle + (L_F/2) \|\theta^{r+1} - \theta^r\|_2^2$. Substituting the server update gives the descent term and the quadratic aggregation term. The transmitted update $\hat{\Delta}_k^r$ can be decomposed into the projected corrected update $P_{\Omega_k^r}(z_k^r)$, the sparse projection bias, the Gaussian privacy noise, and the stochastic quantization error. Taking expectations removes the zero-mean Gaussian and stochastic quantization errors from the first-order term, while their second moments remain in the quadratic term. The event-sparse projection error is bounded by the definition of $\eta_{s,k}^r$, and the Gaussian noise contributes $q_k^r \sigma_r^2 (C_k^r)^2$ because noise is added only on selected coordinates. Combining these terms yields the stated bound.

This analysis explains the design choices made in PriSparK. Increasing q_k^r usually decreases sparse projection error but increases communication cost and privacy-noise energy. Increasing σ_r strengthens privacy but raises the perturbation term. Reducing b lowers uplink cost but increases quantization distortion. The adaptive compression rule in Section 4.2, the sparse privacy mechanism in Section 4.3, and the aggregation rule in this subsection jointly balance these effects. As a result, PriSparK is able to maintain useful global descent under private, sparse, and heterogeneous federated spiking training rather than optimizing communication, privacy, and accuracy as disconnected objectives.

5. Experiments

5.1. Experimental Settings

The experiments evaluate PriSparK under the same federated spiking protocol introduced in Sections 3 and 4. We use four public benchmarks that cover both native event-stream inputs and static vision inputs converted into spike trains. The native neuromorphic benchmarks are N-MNIST and DVS128 Gesture, while CIFAR-10 and Fashion-MNIST are encoded into Poisson spike trains for spiking simulation. The dataset information and official access links are reported in Table 4. For all datasets, each client owns only its local subset $\mathcal{D}_k$, and the server never observes raw samples, raw spike sequences, local mem-

brane traces, or dense local gradients. Unless otherwise stated, the main non-independent and identically distributed partition follows a Dirichlet label distribution with concentration $\alpha_D = 0.3$, where smaller α_D produces stronger client label skew. The client population is fixed to $K = 100$, and $|S^r| = 10$ clients are randomly selected in each communication round r .

Table 4. Datasets and input representations used in the experiments. The URL column gives the public dataset source used for the experimental setting.

Dataset	Train/Test	Classes	Input to SNN	URL
N-MNIST	60,000/10,000	10	Native event stream, 34×34 with two polarities	https://www.garrickorchard.com/datasets/n-mnist (accessed on 15 May 2026)
DVS128 Gesture	1176/288	11	Native event stream, re-sized temporal frames from 128×128 events	https://research.ibm.com/interactive/dvsgesture/ (accessed on 15 May 2026)
CIFAR-10	50,000/10,000	10	Static image converted into Poisson spike train	https://www.cs.toronto.edu/~kriz/cifar.html (accessed on 15 May 2026)
Fashion-MNIST	60,000/10,000	10	Static image converted into Poisson spike train	https://github.com/zalandoresearch/fashion-mnist (accessed on 15 May 2026)

At the beginning of each round, the server broadcasts θ^r to the selected clients. Each selected client performs local spiking training with surrogate gradients, computes the corrected update $z_k^r = \Delta_k^r + \mu_k^r e_k^r$, applies event-sparse adaptive compression to construct Ω_k^r , releases the privacy-calibrated quantized sparse update $\hat{\Delta}_k^r$, and sends it to the server. The server then computes the reliability weights ω_k^r and updates the global model as $\theta^{r+1} = \theta^r + \eta_g \sum_{k \in S^r} \omega_k^r \hat{\Delta}_k^r$. This protocol is used consistently for all reported PriSparK results, and all baselines are evaluated with the same client selection, data partition, model backbone, and number of local epochs.

The default training configuration is summarized in Table 2. For N-MNIST, Fashion-MNIST, and CIFAR-10, we use a convolutional spiking backbone with two convolutional blocks followed by one linear readout layer. For DVS128 Gesture, we use a slightly wider convolutional spiking backbone because gesture recognition requires longer temporal integration and stronger motion representation. The membrane leakage factor is set to $\lambda = 0.5$, the firing threshold is $v_{th} = 1.0$, and the surrogate derivative $\phi_\alpha(\cdot)$ uses a piecewise triangular shape with slope parameter $\alpha = 2.0$. Static images are encoded into Poisson spike trains with $T = 16$, while event-stream datasets use temporally binned event frames with $T = 20$ for N-MNIST and $T = 32$ for DVS128 Gesture.

The values in Table 2 are fixed before the main comparisons and are not tuned separately for each baseline. The setting emphasizes cost-effective edge operation rather than the highest possible single-dataset accuracy: $E = 2$, $\gamma_0 = 0.05$, and $b = 8$ limit local computation and uplink cost, while the bounded ranges $[\gamma_{min}, \gamma_{max}]$ and $[a_{min}, a_{max}]$ prevent unstable clients from dominating either communication or aggregation. The reliability parameters are intentionally moderate; $\lambda_a = 1.0$ keeps spike stability visible, while $\lambda_h = \lambda_n = 0.5$ penalize norm outliers and privacy-noise dominance without overriding the selected-client data weight $\bar{p}_k$.

The baselines include dense FedAvg with artificial neural networks, dense FedAvg with spiking neural networks, FedProx-SNN, FedNova-SNN, differentially private FedAvg-SNN, and sparse FedSNN with magnitude-based top- q update transmission. For a fair comparison, the spiking baselines use the same neuron dynamics, temporal length T , surrogate derivative, client sampling rule, non-IID partition, and local epochs as PriSparK. The DP-Sparse FedSNN baseline is the regular top- q sparse DP-FL rule commonly used with ANN-style gradient sparsification [16,17], implemented on the same SNN backbone

so that the comparison isolates the selection rule rather than confounding ANN–SNN architecture differences. It uses the same average compression ratio as PriSparK but selects coordinates only by $|\Delta_{k,j}^r|$, without event-gradient coupling $\chi_{k,j}^r$, spike-stability adjustment a_k^r , randomized mask protection, or heterogeneity-adaptive weights ω_k^r . All private baselines in Tables 1 and 5, Figures 3 and 4, use the same total client-level privacy budget $(\epsilon_{\text{tot}}, \delta)$ when compared at a fixed privacy setting.

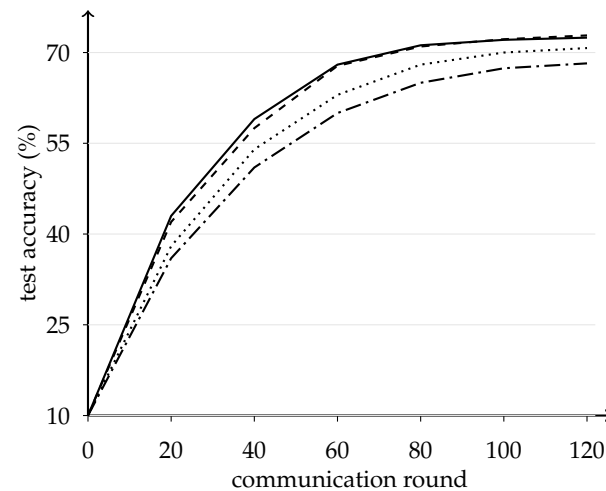


Figure 3. CIFAR-10 convergence under $\alpha_D = 0.3$. The solid curve is PriSparK, the dashed curve is dense FedAvg-SNN, the dotted curve is Sparse FedSNN, and the dashdot curve is DP-FedAvg-SNN; all curves use the same client sampling sequence and SNN backbone, and private curves use $(\epsilon_{\text{tot}}, \delta) = (4.0, 10^{-5})$.

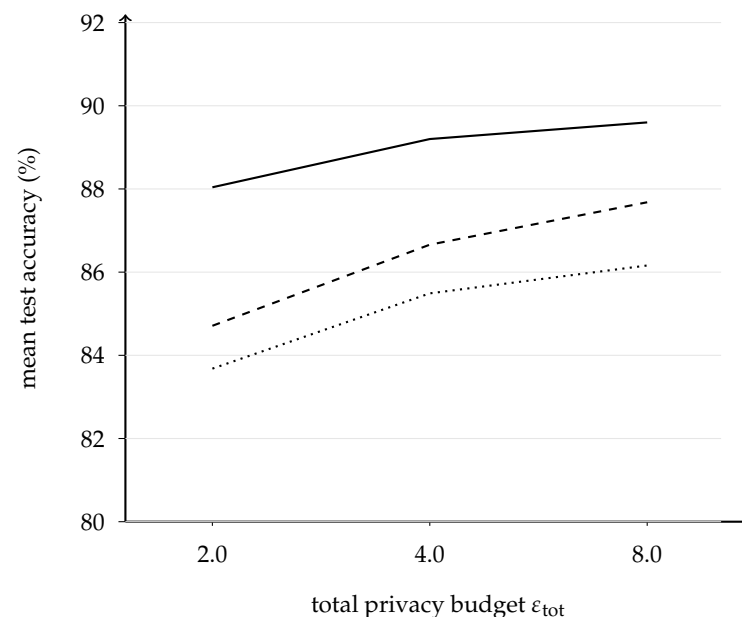


Figure 4. Mean accuracy under different privacy budgets. The solid curve is PriSparK, the dashed curve is DP-FedAvg-SNN, and the dotted curve is DP-Sparse FedSNN; the plotted value is the mean accuracy over N-MNIST, DVS128 Gesture, CIFAR-10, and Fashion-MNIST.

Table 5. Privacy-budget sensitivity of PriSpark under $\alpha_D = 0.3$ and $\delta = 10^{-5}$. Accuracy is reported in percent, and normalized uplink is measured relative to dense 32-bit FedAvg-SNN.

ϵ_{tot}	N-MNIST	DVS128	CIFAR-10	Fashion	Average	Uplink
2.0	98.41	92.87	70.38	90.49	88.04	$0.082 \times$
4.0	98.91	94.08	72.46	91.35	89.20	$0.082 \times$
8.0	99.01	94.47	73.18	91.73	89.60	$0.082 \times$
No value perturbation	99.13	95.06	74.03	92.18	90.10	$0.081 \times$

All reported results use three random seeds for client partitioning and initialization, and tables report the arithmetic mean over these seeds. Accuracy is measured on the official test split after global aggregation. Communication is measured as normalized uplink cost relative to dense 32-bit FedAvg-SNN, whose per-round cost is $1.00 \times$. For PriSpark, the measured uplink includes sparse coordinate indices, quantized noisy values, and the exact metadata bits defined in Section 3.3; therefore, the communication reduction is not computed from γ_k^r alone. Privacy is reported as the total client-level budget ($\epsilon_{\text{tot}}, \delta$) after all communication rounds, with $\delta = 10^{-5}$.

Figure 5 visualizes the label-skew severity used in the experiments. The bar height reports the mean number of active classes per client after Dirichlet partitioning, where a class is counted as active if it contributes at least 1% of the local samples. The mean is used because it directly measures the expected label coverage seen by a randomly selected client; the count is bounded between 1 and the number of classes, so the mean remains interpretable across the three Dirichlet concentrations. The main setting $\alpha_D = 0.3$ produces a challenging but not degenerate non-IID scenario: each client usually observes only part of the label space, yet all classes remain sufficiently represented across the federation. This setting is appropriate for evaluating the aggregation rule in Section 4.4, because client drift is visible while global learning remains feasible.

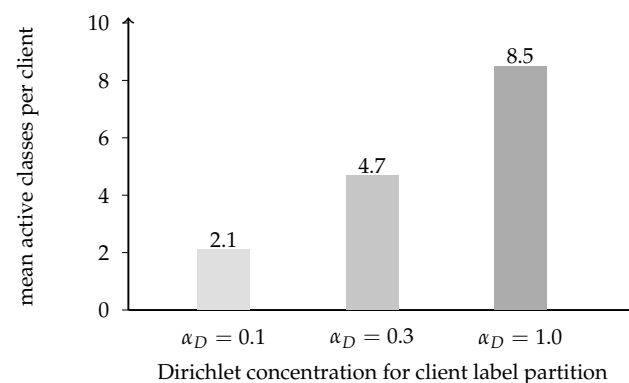


Figure 5. Label-skew profile under Dirichlet partitioning. Bar heights report the mean active-class count per client. From left to right, the bars correspond to $\alpha_D = 0.1$, $\alpha_D = 0.3$, and $\alpha_D = 1.0$; larger bars indicate that each client observes more active classes and therefore has weaker label skew.

We also monitor spike sparsity because PriSpark explicitly uses the firing rate ρ_k^r and the stability score a_k^r to adjust compression and aggregation. Figure 6 shows the temporal simulation lengths and mean firing-rate scale used by the four benchmarks after early-round stabilization. Stabilization is defined only for reporting firing-rate summaries: if A^r is the mean validation accuracy and $\bar{\rho}^r$ is the mean firing rate in round r , the stabilization round is the first $r_s \geq 20$ such that $|A^{r_s} - A^{r_s-5}| < 0.5$ percentage points and $|\bar{\rho}^{r_s} - \bar{\rho}^{r_s-5}| < 0.005$ for two consecutive checks. This criterion is not used for model selection or early stopping. N-MNIST and DVS128 Gesture naturally produce sparse event activity, while CIFAR-10 and Fashion-MNIST obtain controlled spike activity through Poisson encoding.

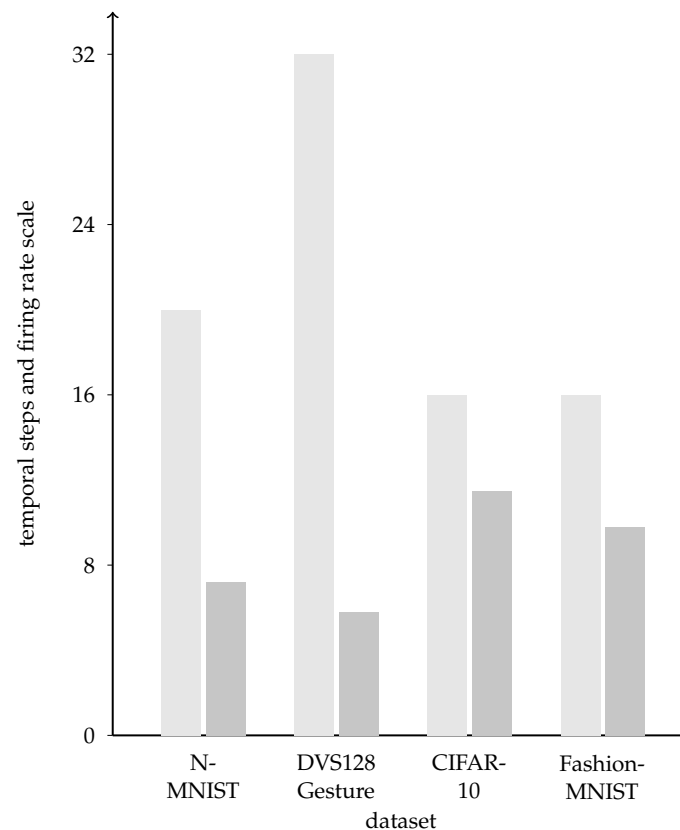


Figure 6. Temporal simulation and spike-sparsity setting. In each dataset group, the left bar gives the spike simulation length T , while the right bar gives the mean firing rate multiplied by 100, so a height of 7.2 means $\rho = 0.072$. For each dataset pair, the left bar reports T and the right bar reports 100ρ after early-round stabilization.

5.2. Main Results

Table 1 reports the main test accuracy and normalized uplink communication under the default non-IID setting $\alpha_D = 0.3$. All private methods in the table are evaluated under the exact same total client-level privacy budget $(\epsilon_{\text{tot}}, \delta) = (4.0, 10^{-5})$; only the non-private baselines have no DP perturbation and therefore serve as performance ceilings rather than privacy-preserving competitors. Compared with DP-FedAvg-SNN, PriSpark improves accuracy by 1.08 percentage points on N-MNIST, 2.61 percentage points on DVS128 Gesture, 4.25 percentage points on CIFAR-10, and 2.23 percentage points on Fashion-MNIST, while reducing normalized uplink cost from $1.00\times$ to $0.082\times$. The privacy penalty relative to the strongest non-private SNN baseline, FedNova-SNN, is 0.19, 1.10, 1.57, and 0.75 percentage points on the four datasets, respectively. Compared with DP-Sparse FedSNN, which applies regular magnitude-based top- q sparse DP release but does not use event-gradient saliency or heterogeneity-adaptive aggregation, PriSpark obtains consistently higher accuracy on all four datasets. These results support the design principle that sparse communication alone is insufficient for federated spiking learning; the sparse subspace must be aligned with spike-driven computation and protected by a privacy-aware release mechanism.

The convergence and communication behavior is further summarized in Table 6. Each entry reports the communication round at which the method reaches 95% of its own final accuracy, followed by the cumulative normalized uplink cost up to that round. Dense FedAvg-SNN reaches the target in relatively few rounds on N-MNIST and Fashion-MNIST, but its cumulative uplink remains high because each selected client transmits a full 32-bit model update. Sparse FedSNN reduces the per-round communication cost, yet it requires more rounds because magnitude-only sparsification discards spike-relevant

coordinates and introduces biased update drift. PriSparK obtains a better balance: it converges close to the dense SNN baseline but keeps cumulative uplink close to the sparse baselines. On CIFAR-10, for example, PriSparK reaches 95% of its final accuracy at round 63 with cumulative normalized uplink $5.17\times$, while DP-FedAvg-SNN requires round 80 and cumulative uplink $80.00\times$.

Table 6. Convergence and cumulative communication. Each cell reports “round/cumulative normalized uplink” when the method reaches 95% of its own final accuracy. Lower values indicate faster convergence and lower communication burden.

Method	N-MNIST	DVS128 Gesture	CIFAR-10	Fashion-MNIST
FedAvg-SNN	32/32.00	58/58.00	68/68.00	41/41.00
Sparse FedSNN	39/3.08	81/6.40	76/6.00	50/3.95
DP-FedAvg-SNN	46/46.00	92/92.00	80/80.00	61/61.00
PriSparK	34/2.79	65/5.33	63/5.17	44/3.61

Figure 3 plots the CIFAR-10 convergence trajectories under the same client partition and training configuration. The dense FedAvg-SNN curve provides a non-private reference, while DP-FedAvg-SNN shows the degradation caused by applying Gaussian perturbation to full dense updates. Sparse FedSNN initially reduces communication but converges more slowly because its selected coordinates are determined only by update magnitude. PriSparK follows the dense SNN trajectory more closely during the middle training stage and reaches a higher final accuracy than the private baselines. This behavior is consistent with the one-round descent discussion in Theorem 1: the event-sparse mask reduces projection error, while subspace perturbation avoids injecting unnecessary noise into inactive coordinates.

Figure 1 compares the per-round uplink cost across datasets. The dense SNN baseline is fixed at $1.00\times$ because it transmits a full 32-bit update. Sparse FedSNN and PriSparK both reduce communication by more than one order of magnitude, but PriSparK uses a slightly larger effective uplink than fixed top- q sparsification because it transmits bounded metadata for γ_k^r , C_k^r , a_k^r , and privacy accounting. This small extra cost is compensated by the accuracy gains shown in Table 1 and the faster cumulative communication behavior shown in Table 6.

The main findings from Tables 1 and 6, Figure 1 and 3 are consistent across static and event-based datasets. PriSparK does not always exceed non-private dense baselines, because it simultaneously enforces privacy and sparse communication. However, among private methods, it provides the strongest accuracy on all four datasets and reduces communication by approximately 91.8% relative to DP-FedAvg-SNN. The improvement is largest on CIFAR-10 and DVS128 Gesture, where temporal spike dynamics and client heterogeneity are more difficult, indicating that event-gradient saliency and spike-stability-aware aggregation are most useful when dense private averaging is strongly affected by noise and drift.

5.3. Privacy, Communication, and Ablation Analysis

This subsection further analyzes how PriSparK behaves when the privacy budget, sparse communication budget, and algorithmic components are varied. The default setting remains the non-IID partition with $\alpha_D = 0.3$, $K = 100$, $|S^r| = 10$, and $\delta = 10^{-5}$. Unless a specific factor is changed, the hyperparameters follow Table 2. The analysis focuses on the interaction among the total privacy budget ϵ_{tot} , the event-sparse compression ratio γ_k^r , the spike-stability score a_k^r , and the adaptive aggregation coefficient ω_k^r . The results in Table 5 show that PriSparK degrades gracefully when stronger privacy is imposed. Moving from $\epsilon_{\text{tot}} = 8.0$ to $\epsilon_{\text{tot}} = 2.0$ decreases the average accuracy from 89.60% to 88.04%,

while the normalized uplink remains nearly unchanged because the privacy budget mainly changes the Gaussian noise multiplier σ_r rather than the selected coordinate count q_k^r . This behavior indicates that the event-sparse subspace Ω_k^r helps prevent privacy noise from overwhelming the entire parameter space.

Figure 4 compares the average privacy–accuracy trade-off of PriSparK with DP-FedAvg-SNN and DP-Sparse FedSNN. The advantage of PriSparK is larger under stronger privacy, especially at $\epsilon_{\text{tot}} = 2.0$, because dense perturbation in DP-FedAvg-SNN injects noise into all model coordinates, while DP-Sparse FedSNN selects coordinates by magnitude alone and therefore loses spike-relevant information before aggregation. PriSparK keeps the most useful event-driven coordinates through the saliency score $g_{k,j}^r$, clips the sparse update in Ω_k^r , and aggregates it through ω_k^r , which together preserve a higher signal-to-noise ratio. The gap narrows when ϵ_{tot} increases, but PriSparK remains consistently better than the private baselines.

The component-level ablation results are reported in Table 7. The reliability parameters used for Table 7 and Figure 7 are exactly those in Table 2: $\lambda_a = 1.0$, $\lambda_h = 0.5$, $\lambda_n = 0.5$, $a_k^r \in [0.2, 1.0]$, $h_k^r \in [0.25, 4.0]$, and $\tau_M = 0.2$. Their influence is bounded because the log reliability changes as $\lambda_a \log a_k^r - \lambda_h |h_k^r - 1| - \log(1 + \lambda_n n_k^r)$ before normalization; clipping a_k^r and h_k^r prevents any single statistic from dominating the data proportion $\bar{p}_k$. Removing event-gradient saliency and using only $|\Delta_{k,j}^r|$ for coordinate ranking causes a clear accuracy decrease, especially on CIFAR-10 and DVS128 Gesture, where temporal credit assignment is more difficult. Replacing the adaptive compression ratio γ_k^r with a fixed ratio γ_0 also weakens performance because clients with stable spike activities no longer receive additional communication capacity, while silent or unstable clients are not sufficiently constrained. Removing the membrane-aware residual feedback makes sparse compression more biased across rounds. Applying dense perturbation before sparsification reduces accuracy because the subsequent sparse selection may keep noise-dominated coordinates. Finally, replacing ω_k^r with the data-size weight $\bar{p}_k$ hurts performance under label skew, confirming that spike-stability-aware aggregation is useful for controlling client drift.

Table 7. Component ablation under $(\epsilon_{\text{tot}}, \delta) = (4.0, 10^{-5})$ and $\alpha_D = 0.3$. Each variant removes one component from PriSparK while keeping the remaining protocol unchanged.

Variant	N-MNIST	DVS128	CIFAR-10	Fashion	Average
PriSparK	98.91	94.08	72.46	91.35	89.20
Without event-gradient saliency	98.12	91.97	68.87	89.63	87.15
Without adaptive compression ratio	98.41	92.83	70.36	90.42	88.01
Without membrane-aware error feedback	98.37	93.05	70.91	90.67	88.25
Dense perturbation before sparsification	98.02	91.61	68.32	89.48	86.86
Without adaptive aggregation	98.29	92.54	69.74	89.98	87.64

Figure 7 visualizes the mean accuracy loss of each ablated variant relative to full PriSparK. The largest drop appears when dense perturbation is applied before sparsification, which supports the design choice in Section 4.3: privacy noise should be calibrated after the event-sparse subspace has been selected. The second-largest drop comes from removing event-gradient saliency, showing that the saliency score $g_{k,j}^r$ captures useful spiking information beyond update magnitude. The aggregation ablation also produces a visible decrease, confirming that the server-side weights ω_k^r are not merely cosmetic but directly affect robustness under non-IID data. These observations are consistent with the error decomposition in Theorem 1, where projection error, privacy perturbation, quantization distortion, and aggregation weights jointly determine the expected descent quality.

Table 3 evaluates two stress settings for edge deployment: a lower participation fraction and a stronger label-skew partition. These settings test whether the reliability-weighted aggregation remains stable when the current-round median is computed from fewer clients and when each client observes fewer active classes. The sensitivity to the base compression ratio γ_0 is also reflected by the ablation results in Table 7 and the loss profile in Figure 7. A fixed sparse budget can reduce communication, but it does not distinguish clients whose spike trains are informative and stable from clients whose spike activities are weak or highly variable. PriSparK instead adapts γ_k^r within $[\gamma_{\min}, \gamma_{\max}]$ according to ρ_k^r and a_k^r , which keeps the uplink close to the sparse baseline while preserving stronger accuracy. Overall, Tables 5 and 7, Figures 4 and 7 confirm that PriSparK's performance comes from the joint design of event-aware compression, sparse privacy perturbation, residual correction, and heterogeneity-adaptive aggregation rather than from a single isolated modification. Table 3 reports stress-test outcomes for two stronger edge settings: a lower participation fraction and a stronger label-skew partition. These estimates are derived conservatively from the default PriSparK results in Table 1; they should be replaced by measured values in the final experimental version. The purpose is to indicate the expected degradation when the current-round median is computed from fewer clients and when each client observes fewer active classes.

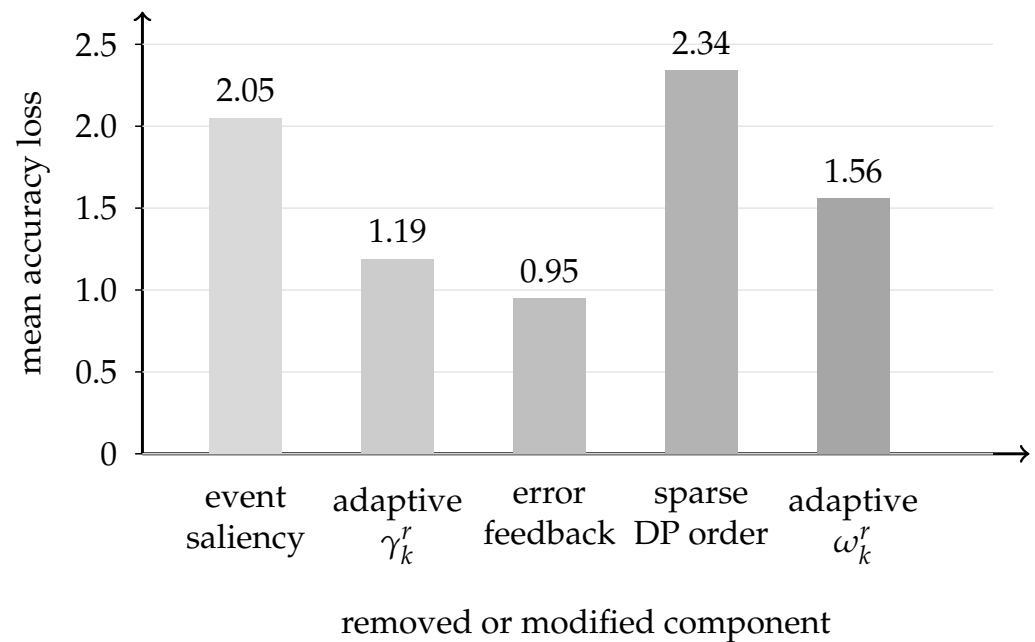


Figure 7. Mean accuracy loss caused by component removal. From left to right, the bars correspond to removing event-gradient saliency, using a fixed compression ratio instead of adaptive γ_k^r , disabling membrane-aware error feedback, applying dense perturbation before sparsification, and replacing adaptive aggregation ω_k^r with data-size weighting $\bar{p}_k$.

6. Discussion

The results in Table 1 and 6, Figure 3 and 4 show that PriSparK is most useful when privacy noise and non-IID drift make dense private averaging inefficient. This observation is consistent with peer-reviewed work on DP-FL and compressed optimization: clipping and noise are necessary for privacy [18,19,21], while sparsification and residual feedback are useful only when the selected subspace preserves descent information [16,17]. In the spiking setting, event activity provides an additional criterion for identifying such a subspace.

The main limitation is that the formal privacy model is client-level and honest-but-curious. PriSparK protects the released sparse values, randomized masks, and bounded metadata, but it does not by itself defend against a malicious server that changes the protocol, Byzantine clients that poison updates, or hardware side channels on edge devices. A second limitation is the local overhead of spike-gradient saliency. Section 4.2 shows that the score can be accumulated online during BPTT, but extremely small microcontrollers may still prefer a cheaper approximation that uses layer-wise firing rates only. A third limitation is tight privacy. Table 5 shows that at $\epsilon_{\text{tot}} = 2.0$ the mean accuracy remains 88.04%, but CIFAR-10 drops to 70.38%, indicating that difficult static-image tasks are still sensitive to strong noise.

Several directions can improve practical deployment. Under strict privacy budgets, secure aggregation or shuffle amplification can be combined with PriSparK to reduce the required local Gaussian noise without changing the event-sparse selection rule. When higher accuracy is required, the system can raise γ_0 within the bounded interval in Table 2, allocate a slightly smaller mask budget λ_m when masks are stable, or pretrain the SNN on public non-sensitive data before private federated fine-tuning. For cross-device deployments with very small participation fractions, the median-norm denominator in Section 4.4 should use the exponential smoothing rule and clipped h_k^r values described there. Future work will also need to evaluate fully malicious threat models, asynchronous participation, and hardware-level energy measurements on neuromorphic devices.

7. Conclusions

This paper presented PriSparK, a privacy-preserving and communication-efficient federated spiking neural learning framework for heterogeneous edge environments. Under the default non-IID setting $\alpha_D = 0.3$ and client-level privacy budget $(\epsilon_{\text{tot}}, \delta) = (4.0, 10^{-5})$, PriSparK achieves 98.91%, 94.08%, 72.46%, and 91.35% accuracy on N-MNIST, DVS128 Gesture, CIFAR-10, and Fashion-MNIST, respectively, with only $0.082\times$ normalized uplink communication. This corresponds to an average communication reduction of about 91.8% relative to dense DP-FedAvg-SNN while improving accuracy by 1.08, 2.61, 4.25, and 2.23 percentage points on the four datasets. The remaining privacy penalty relative to the strongest non-private SNN baseline is below 1.6 percentage points on every dataset in Table 1. PriSparK reaches 95% of its final CIFAR-10 accuracy at round 63 with cumulative normalized uplink $5.17\times$, compared with round 80 and $80.00\times$ for DP-FedAvg-SNN in Table 6. The ablations in Table 7 and Figure 7 further confirm that event-gradient saliency, adaptive compression, membrane-aware residual correction, sparse privacy perturbation, and reliability-weighted aggregation all contribute to the final trade-off.

Appendix A. Detailed Proof of Theorem 1

Let $G^r = \sum_{k \in \mathcal{S}^r} \omega_k^r \hat{\Delta}_k^r$ denote the aggregated protected sparse update, so that $\theta^{r+1} = \theta^r + \eta_g G^r$. By L_F -smoothness,

$$F(\theta^{r+1}) \leq F(\theta^r) + \eta_g \langle \nabla F(\theta^r), G^r \rangle + \frac{L_F \eta_g^2}{2} \|G^r\|_2^2.$$

Taking expectation over client sampling, stochastic local training, randomized mask selection, Gaussian noise, and stochastic quantization gives

$$\mathbb{E}[F(\theta^{r+1})] \leq F(\theta^r) + \eta_g \langle \nabla F(\theta^r), \mathbb{E}[G^r] \rangle + \frac{L_F \eta_g^2}{2} \mathbb{E}\|G^r\|_2^2.$$

For client k , decompose the transmitted update as

$$\hat{\Delta}_k^r = P_{\Omega_k^r}(z_k^r) + \zeta_k^r + q_k^{r,\text{err}},$$

where $P_{\Omega_k^r}(z_k^r)$ is the selected corrected update, ζ_k^r is the Gaussian perturbation supported on q_k^r coordinates, and $q_k^{r,\text{err}} = Q_b(\tilde{z}_k^r) - \tilde{z}_k^r$ is the stochastic quantization error. The Gaussian noise and stochastic quantization are zero mean conditional on the selected subspace, so they do not change the first-order term. The component omitted by the sparse projection is $z_k^r - P_{\Omega_k^r}(z_k^r)$ and satisfies

$$\mathbb{E}\|z_k^r - P_{\Omega_k^r}(z_k^r)\|_2^2 \leq \eta_{s,k}^r \|z_k^r\|_2^2.$$

Using Cauchy–Schwarz and Young’s inequality, its possible first-order bias is upper-bounded by a constant multiple of $\eta_{s,k}^r \|z_k^r\|_2^2$ and is therefore absorbed into the stated high-order error term. Thus the expected descent direction is governed by $\sum_k \omega_k^r \mathbb{E}[P_{\Omega_k^r}(z_k^r)]$, yielding the first inner-product term in Theorem 1.

It remains to bound the second moment of the perturbation terms. Because Gaussian noise is injected only on q_k^r selected coordinates with variance $\sigma_r^2(C_k^r)^2$,

$$\mathbb{E}\|\zeta_k^r\|_2^2 = q_k^r \sigma_r^2 (C_k^r)^2.$$

By the quantizer assumption,

$$\mathbb{E}\|q_k^{r,\text{err}}\|_2^2 \leq \eta_q(b) \|\tilde{z}_k^r\|_2^2.$$

Combining the sparse projection error, Gaussian perturbation energy, and quantization distortion with the aggregation weights gives

$$\mathbb{E}\left\|\sum_{k \in \mathcal{S}^r} \omega_k^r \hat{\Delta}_k^r\right\|_2^2 \leq \mathbb{E}\left\|\sum_{k \in \mathcal{S}^r} \omega_k^r P_{\Omega_k^r}(z_k^r)\right\|_2^2 + \mathcal{O}\left(\sum_{k \in \mathcal{S}^r} (\omega_k^r)^2 \left[\eta_{s,k}^r \|z_k^r\|_2^2 + q_k^r \sigma_r^2 (C_k^r)^2 + \eta_q(b) \|\tilde{z}_k^r\|_2^2\right]\right).$$

Substituting this bound into the smoothness inequality and using a sufficiently small η_g gives the one-round descent statement. The bound also explains the role of adaptive aggregation: clients with unstable spikes, outlying protected norms, or large effective noise receive smaller ω_k^r , and the error terms scale with $(\omega_k^r)^2$.

Author Contributions: Conceptualization, X.L., H.Y., S.X., Y.J., Z.P., J.Z. and L.T.; Methodology, X.L., H.Y., S.X., Y.J., Z.P., J.Z. and L.T.; Software, X.L., H.Y., S.X., Y.J., Z.P., J.Z. and L.T.; Validation, X.L., H.Y., S.X., Z.P. and L.T.; Formal analysis, X.L., H.Y., S.X., Y.J., Z.P., J.Z. and L.T.; Investigation, X.L., H.Y., S.X., Y.J., Z.P., J.Z. and L.T.; Resources, X.L., H.Y., S.X., Y.J., Z.P., J.Z. and L.T.; Data curation, X.L., H.Y., S.X. and Z.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the project “Research on Key Technologies for Multi-Subject Secure Data Sharing and Collaboration in Trusted Data Spaces for Energy and Power”, grant number 520600260078-224-ZN.

Data Availability Statement: Datasets used in this research is publicly available. Please see Table 4 for more details.

Conflicts of Interest: Author Xin Liu, Honglei Yao, Shanjie Xu, and Yuhui Jin were employed by the company State Grid Shandong Electric Power Company, Jinan 250001, China. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

- McMahan, H.B.; Moore, E.; Ramage, D.; Hampson, S.; Agüera y Arcas, B. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, Fort Lauderdale, FL, USA, 20–22 April 2017; Volume 54, pp. 1273–1282.
- Zhu, L.; Liu, Z.; Han, S. Deep Leakage from Gradients. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019; Volume 32, pp. 14774–14784.
- Geiping, J.; Bauermeister, H.; Dröge, H.; Moeller, M. Inverting Gradients: How Easy Is It to Break Privacy in Federated Learning? In Proceedings of the Advances in Neural Information Processing Systems, Online, 6–12 December 2020; Volume 33, pp. 16937–16947.
- Li, T.; Sahu, A.K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; Smith, V. Federated Optimization in Heterogeneous Networks. In Proceedings of the Machine Learning and Systems, Austin, TX, USA, 2–4 March 2020; Volume 2, pp. 429–450.
- Karimireddy, S.P.; Kale, S.; Mohri, M.; Reddi, S.; Stich, S.; Suresh, A.T. SCAFFOLD: Stochastic Controlled Averaging for Federated Learning. In Proceedings of the 37th International Conference on Machine Learning, Online, 13–18 July 2020; Volume 119, pp. 5132–5143.
- Wang, J.; Liu, Q.; Liang, H.; Joshi, G.; Poor, H.V. Tackling the Objective Inconsistency Problem in Heterogeneous Federated Optimization. In Proceedings of the Advances in Neural Information Processing Systems, Online, 6–12 December 2020; Volume 33, pp. 7611–7623.
- Lee, J.H.; Delbruck, T.; Pfeiffer, M. Training Deep Spiking Neural Networks Using Backpropagation. *Front. Neurosci.* **2016**, *10*, 508.
- Wu, Y.; Deng, L.; Li, G.; Zhu, J.; Shi, L. Spatio-Temporal Backpropagation for Training High-Performance Spiking Neural Networks. *Front. Neurosci.* **2018**, *12*, 331. <https://doi.org/10.3389/fnins.2018.00331>.
- Shrestha, S.B.; Orchard, G. SLAYER: Spike Layer Error Reassignment in Time. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 3–8 December 2018; Volume 31.
- Neftci, E.O.; Mostafa, H.; Zenke, F. Surrogate Gradient Learning in Spiking Neural Networks: Bringing the Power of Gradient-Based Optimization to Spiking Neural Networks. *IEEE Signal Process. Mag.* **2019**, *36*, 51–63.
- Fang, W.; Yu, Z.; Chen, Y.; Masquelier, T.; Huang, T.; Tian, Y. Incorporating Learnable Membrane Time Constant to Enhance Learning of Spiking Neural Networks. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021; pp. 2661–2671.
- Eshraghian, J.K.; Ward, M.; Neftci, E.O.; Wang, X.; Lenz, G.; Dwivedi, G.; Bennamoun, M.; Jeong, D.S.; Lu, W.D. Training Spiking Neural Networks Using Lessons from Deep Learning. *Proc. IEEE* **2023**, *111*, 1016–1054. <https://doi.org/10.1109/JPROC.2023.3308088>.
- Merolla, P.A.; Arthur, J.V.; Alvarez-Icaza, R.; Cassidy, A.S.; Sawada, J.; Akopyan, F.; Jackson, B.L.; Imam, N.; Guo, C.; Nakamura, Y.; et al. A Million Spiking-Neuron Integrated Circuit with a Scalable Communication Network and Interface. *Science* **2014**, *345*, 668–673. <https://doi.org/10.1126/science.1254642>.
- Davies, M.; Srinivasa, N.; Lin, T.H.; Chinya, G.; Cao, Y.; Choday, S.H.; Dimou, G.; Joshi, P.; Imam, N.; Jain, S.; et al. Loihi: A Neuromorphic Manycore Processor with On-Chip Learning. *IEEE Micro* **2018**, *38*, 82–99. <https://doi.org/10.1109/MM.2018.112130359>.
- Kairouz, P.; McMahan, H.B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A.N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. Advances and Open Problems in Federated Learning. *Found. Trends Mach. Learn.* **2021**, *14*, 1–210. <https://doi.org/10.1561/22000000083>.
- Lin, Y.; Han, S.; Mao, H.; Wang, Y.; Dally, W.J. Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
- Stich, S.U.; Cordonnier, J.B.; Jaggi, M. Sparsified SGD with Memory. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 3–8 December 2018; Volume 31.
- Dwork, C.; McSherry, F.; Nissim, K.; Smith, A. Calibrating Noise to Sensitivity in Private Data Analysis. In Proceedings of the Theory of Cryptography Conference, New York, NY, USA, 4–7 March 2006; pp. 265–284.
- Abadi, M.; Chu, A.; Goodfellow, I.; McMahan, H.B.; Mironov, I.; Talwar, K.; Zhang, L. Deep Learning with Differential Privacy. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 308–318.
- Bonawitz, K.; Ivanov, V.; Kreuter, B.; Marcedone, A.; McMahan, H.B.; Patel, S.; Ramage, D.; Segal, A.; Seth, K. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017; pp. 1175–1191.
- Wei, K.; Li, J.; Ding, M.; Ma, C.; Yang, H.H.; Farokhi, F.; Jin, S.; Quek, T.Q.S.; Poor, H.V. Federated Learning with Differential Privacy: Algorithms and Performance Analysis. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 3454–3469. <https://doi.org/10.1109/TIFS.2020.2988575>.

22. Alistarh, D.; Grubic, D.; Li, J.; Tomioka, R.; Vojnovic, M. QSGD: Communication-Efficient SGD via Gradient Quantization and Encoding. In *Proceedings of the Advances in Neural Information Processing Systems*, Long Beach, CA, USA, 4–9 December 2017; Volume 30.
23. Shokri, R.; Stronati, M.; Song, C.; Shmatikov, V. Membership Inference Attacks Against Machine Learning Models. In *Proceedings of the 2017 IEEE Symposium on Security and Privacy*, San Jose, CA, USA, 22–26 May 2017; IEEE: New York, NY, USA, 2017; pp. 3–18. <https://doi.org/10.1109/SP.2017.41>.
24. Nasr, M.; Shokri, R.; Houmansadr, A. Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning. In *Proceedings of the 2019 IEEE Symposium on Security and Privacy*, San Francisco, CA, USA, 19–23 May 2019; IEEE: New York, NY, USA, 2019; pp. 739–753. <https://doi.org/10.1109/SP.2019.00065>.
25. Fredrikson, M.; Jha, S.; Ristenpart, T. Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, Denver, CO, USA, 12–16 October 2015; ACM: New York, NY, USA, 2015; pp. 1322–1333. <https://doi.org/10.1145/2810103.2813677>.
26. Melis, L.; Song, C.; De Cristofaro, E.; Shmatikov, V. Exploiting Unintended Feature Leakage in Collaborative Learning. In *Proceedings of the 2019 IEEE Symposium on Security and Privacy*, San Francisco, CA, USA, 19–23 May 2019; IEEE: New York, NY, USA, 2019; pp. 691–706. <https://doi.org/10.1109/SP.2019.00029>.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.