TEMPORAL EVIDENCE BEYOND MAGNITUDE FOR COMMUNICATION-EFFICIENT FEDERATED SPIKING LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Federated learning with spiking neural networks (SNNs) offers a promising route to energy-efficient edge intelligence, but communication remains a primary bottleneck. Existing compressors largely inherit an artificial neural network (ANN)-oriented view, ranking updates for transmission by their accumulated magnitude. In SNNs, however, each synaptic update accumulates learning contributions across timesteps under stateful leaky integrate-and-fire (LIF) dynamics, so similar update magnitudes can correspond to markedly different temporal contribution patterns. Magnitude-only ranking cannot distinguish these patterns. We introduce LIF temporal evidence, derived from the alignment between timestep-wise learning signals obtained from a single diagnostic replay and the accumulated local update. Building on this evidence, we propose FedBLADE (*Block-wise LIF-Aware Dynamic Encoding*), which aggregates temporal evidence within architecture- and magnitude-matched blocks, combines it with block update magnitude to prioritize blocks, and admits the highest-ranked blocks under an explicit bit budget, followed by within-block Top-k sparsification, low-bit quantization, and compact relative-position coding. Experiments on Fashion-MNIST, CIFAR-10, and CIFAR-100 under Dirichlet non-IID partitions and uplink budgets of 1–10% of dense transmission show that FedBLADE outperforms representative compression baselines across the tested settings; it also outperforms FedSNN on CIFAR-10 and Fashion-MNIST and achieves competitive performance against several representative federated SNN methods.

1 INTRODUCTION

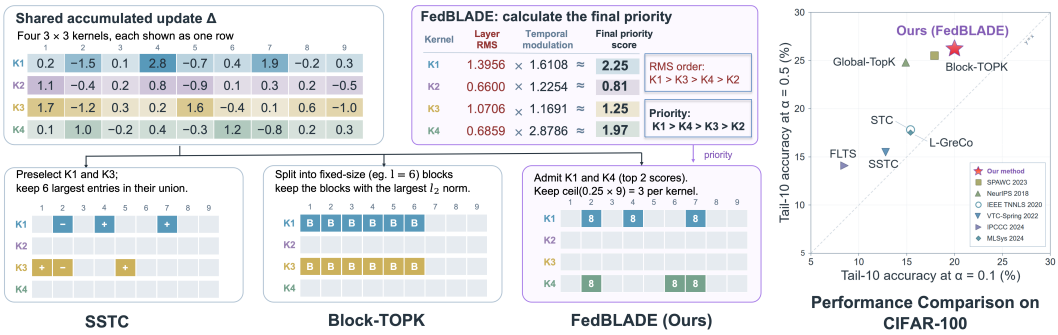


Figure 1: Given the shared accumulated update Δ, our method is able to allocate the communication budget according to temporal importance beyond static magnitude. (The colored cells denote the coordinates retained for communication.) FedBLADE computes the final priority of each kernel by multiplying its layer RMS with a LIF-derived temporal factor, admits the highest-priority kernels, and then sparsifies cells within them, which enforces temporally consistent budget allocation. It demonstrates state-of-the-art tail-10 accuracy on CIFAR-100 under both $\alpha = 0.1$ and $\alpha = 0.5$, while maintaining the same communication budget as its counterparts.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Spiking neural networks (SNNs) are brain-inspired models that represent and process information through discrete spikes over time Maass (1997); Fang et al. (2023). Unlike the memoryless rectified linear unit (ReLU), the widely used leaky integrate-and-fire (LIF) neuron maintains a stateful membrane potential that integrates incoming signals with temporal decay, emits a spike when the potential reaches a threshold, and then resets Fang et al. (2021). This sparse and event-driven computation can substantially reduce energy consumption on neuromorphic hardware, making SNNs a promising route to energy-efficient intelligence on resource-constrained edge and Internet of Things (IoT) devices Li et al. (2026). These characteristics make SNNs particularly appealing for federated learning across distributed edge devices, where models can be trained collaboratively without centralizing local data. However, edge and IoT environments often operate under constrained communication bandwidth, making the repeated exchange of model updates a critical bottleneck for federated SNN training Nguyen et al. (2024); Xie et al. (2024b).

To reduce communication, existing methods compress gradients or model updates through sparsification, quantization, error feedback, structured selection, and joint sparsity–precision optimization Alistarh et al. (2017); Karimireddy et al. (2019); Reisizadeh et al. (2020); Xie et al. (2024a). A large class of sparsification-based compressors nevertheless relies on update magnitude to prioritize transmission, with refinements such as residual correction Lin et al. (2018); Sattler et al. (2019), structured or block-wise selection Mora et al. (2022); Su et al. (2023), layer-wise allocation Markov et al. (2024), and adaptive sparsity Ruan et al. (2024); Vaishnav et al. (2025); Zhang et al. (2025). In SNNs, synaptic gradients are formed through learning signals propagated across timesteps under stateful LIF dynamics Wu et al. (2018); Fang et al. (2021). The accumulated local update collapses timestep-wise LIF learning signals into a single parameter change, leaving their temporal structure unrepresented. Updates with similar magnitudes can therefore have different temporal contribution patterns. This raises the question of how timestep-wise learning signals can be incorporated into transmission priorities alongside update magnitude.

We introduce *LIF temporal evidence* as an auxiliary signal for block prioritization. Before local training, a single diagnostic replay extracts timestep-wise weight-gradient contributions from the round-start model. After local training, their normalized strength and absolute cosine similarity to the accumulated local update determine the evidence assigned to each block. The replay takes no optimizer step, and the evidence is used only to set the transmission order. FedBLADE (*Block-wise LIF-Aware Dynamic Encoding*) incorporates this evidence into a hierarchical block compressor. It partitions model updates into architecture-aligned blocks and uses temporal evidence to modulate layer-normalized block RMS. The encoder visits blocks in descending priority and admits records that fit the coded-bit budget. Within each admitted block, it combines Top-k sparsification, INT8 quantization, and compact relative-position coding. The server uses the same codec with raw-RMS block ranking and downlink error feedback. Our contributions are as follows.

- We identify a limitation of magnitude-only communication ranking for federated SNNs: the accumulated local update collapses learning signals produced across LIF timesteps into a single parameter change. To expose information absent from the accumulated magnitude, we derive block-level temporal evidence from timestep-wise diagnostic gradients and their correspondence with the final local update.

- We propose **FedBLADE** (*Block-wise LIF-Aware Dynamic Encoding*), a hierarchical block compressor that turns temporal evidence into an explicit block-priority signal, selecting and encoding only the most informative blocks within a coded-bit budget so that the communicated update preserves its learning-relevant components. To the best of our knowledge, this is the first communication-compression method for federated SNNs that leverages LIF temporal evidence for block prioritization.

- We conduct experiments on Fashion-MNIST, CIFAR-10, and CIFAR-100 across multiple Dirichlet concentrations and communication budgets. FedBLADE achieves the highest mean accuracy among the evaluated compressors in all reported CIFAR-10 and CIFAR-100 settings and remains competitive on Fashion-MNIST. Component comparisons further show that block-local representation and downlink residual correction are major contributors under tight communication budgets.

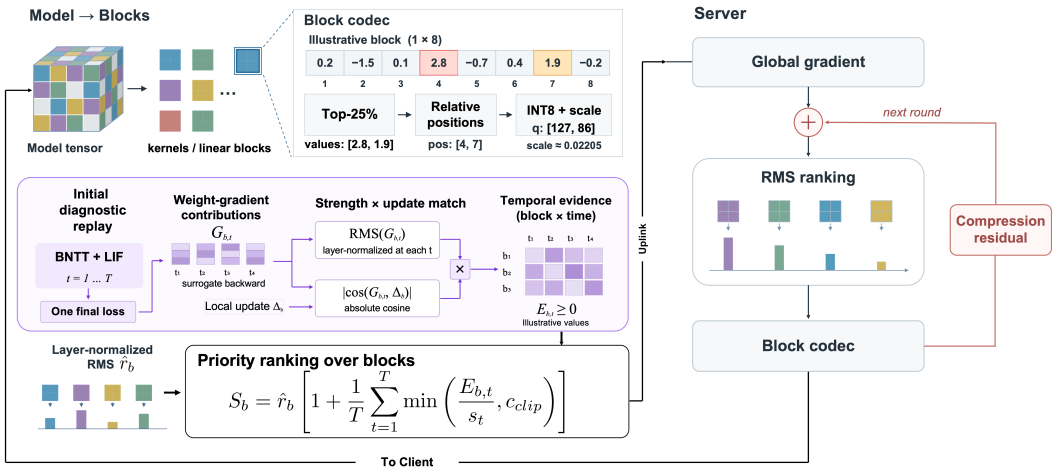


Figure 2: Overview of FedBLADE. Each client partitions its accumulated update into architecture-aligned blocks. An initial diagnostic replay produces timestep-wise weight-gradient contributions, whose normalized strength and absolute alignment with the accumulated update form temporal evidence. This evidence modulates layer-normalized block RMS to determine the uplink block priority, while each visited block is compressed by Top-25% sparsification, INT8 quantization, and compact support coding. The server aggregates decoded updates, adds the previous compression residual to the current global update, and applies Plain-RMS ranking with the same block codec. The new residual is carried to the next round, and the compressed update is returned to clients.

2 Related Work

Magnitude-based compression. Magnitude-based sparsification retains large gradient or update coordinates to reduce transmission cost. Deep Gradient Compression combines this principle with momentum correction and local gradient accumulation, while sparsified SGD with memory compensates for omitted updates across iterations (Lin et al., 2018; Stich et al., 2018). STC combines magnitude-based Top-k selection, ternary quantization, bidirectional residual correction, and Golomb-coded position differences (Sattler et al., 2019). SSTC exploits the concentration of large convolutional updates within kernels: it preselects kernels by their mean absolute update, applies STC over the retained kernel subset, and encodes kernel indices with ternary sign maps (Mora et al., 2022). BlockTOPK instead ranks equal-length blocks by squared ℓ_2 norm and retains complete blocks, jointly selecting their number and quantization precision under a communication constraint (Su et al., 2023). JointSQ further unifies sparsification and quantization by treating omission as zero-bit quantization and optimizing mixed-precision allocation (Xie et al., 2024a). CocktailSGD composes random sparsification, Top-k selection, and quantization to support distributed foundation-model fine-tuning over slow networks (Wang et al., 2023a). At the model level, FedMef combines magnitude-based weight pruning with budget-aware extrusion and scaled activation pruning to support memory-efficient federated dynamic pruning (Huang et al., 2024).

Adaptive Top-k varies the number of retained coordinates across iterations (Ruan et al., 2024), while the Top-k instantiation of L-GreCo allocates sparsity across layers under a compression-error constraint (Markov et al., 2024). FL-CATE adapts sparsification to communication costs using residual-corrected updates (Vaishnav et al., 2025). FedGCC sparsifies accumulated local gradients by magnitude while incorporating error feedback, gradient tracking, and correlation-aware aggregation (Zhang et al., 2025). Clover integrates Top-k sparsification with secure aggregation and differential privacy (Xu et al., 2025).

The update compressors above make compression decisions from gradients or accumulated updates and, where applicable, optimizer or residual state. Their selection and allocation rules operate at coordinate, block, or layer granularity. FedBLADE additionally uses timestep-wise diagnostic signals to modulate layer-normalized block magnitude before budgeted block admission.

Federated spiking neural networks. Federated SNN research has progressively addressed distributed training, statistical heterogeneity, and resource-aware coordination. FedSNN establishes federated training of SNNs and studies its sensitivity to heterogeneous data, stragglers, and gradient noise (Venkatesha et al., 2021). LFNL uses leader election to coordinate neuromorphic model aggregation without a fixed central server (Yang et al., 2022). FedLEC targets label skew through intra-client label-weight calibration and inter-client knowledge distillation (Yu et al., 2024). Other methods exploit spike activity for client selection: SFedCA assigns client credits from changes in firing intensity before and after local training (Zhan et al., 2025), while FedILSR uses integrated layer-wise spike rates to guide energy-aware client participation in wireless federated SNNs (Wang et al., 2026). SFedHIFI supports clients with different model widths through channel-wise matrix decomposition and firing-rate-guided heterogeneous information fusion (Tao et al., 2026). Federated methods such as FedRS, FedLC, DivFL, and FedMoS provide complementary mechanisms for label skew, representative client selection, or optimization under heterogeneous participation (Li & Zhan, 2021; Zhang et al., 2022; Balakrishnan et al., 2022; Wang et al., 2023b). These approaches primarily address local learning, client participation, or model aggregation, while FedBLADE focuses on *which parts* of a participating client's SNN update are communicated.

Among methods that directly target communication efficiency in federated SNNs, FLTS exploits the observed robustness of SNNs to communication perturbations and applies magnitude-based Top-κ sparsification to both uplink and downlink transmission (Nguyen et al., 2024). HDSFL instead replaces full-model exchange with knowledge transfer, communicating hint-layer representations and losslessly compressed spike tensors and aggregating the transmitted knowledge according to client confidence (Xie et al., 2024b).

3 METHOD

FedBLADE separates block prioritization from block encoding. Each client combines its accumulated update with timestep-wise signals from a diagnostic replay to rank architecture-aligned blocks. The encoder then selects the block records and within-block supports that fit the communication budget. The server averages decoded uploads and compresses the downlink using raw-RMS ranking and error feedback. Figure 2 summarizes the pipeline.

3.1 ARCHITECTURE-ALIGNED UPDATE BLOCKS

At round r, a set $\mathcal{C}_r$ of B clients is sampled from N registered clients. Each selected client starts from the shared reconstruction $\mathbf{b}_r$ and obtains weights w_i^r after local training. Its accumulated local update is

$$\Delta_i^r = w_i^r - \mathbf{b}_r. \tag{1}$$

FedBLADE partitions Δ_i^r into fixed architecture-aligned blocks. For the architectures considered here, convolutional weights are grouped by input–output channel kernels, while linear rows are split along the input dimension into contiguous segments of up to 16 weights. Let $\mathcal{B}$ denote the set of $M = |\mathcal{B}|$ blocks, and let n_b and $\ell(b)$ denote the size and layer of block b. The blocks in layer ℓ form $\mathcal{B}_\ell = \{b \in \mathcal{B} : \ell(b) = \ell\}$. The block structure is fixed across rounds, while transmission priorities are recomputed from each local update. For a fixed client and round, we write $\Delta = \Delta_i^r$ and let Δ_b denote its restriction to block b. All update and state vectors below refer to the compressible convolutional and linear weights; normalization state is handled separately.

Given a block order π, the encoder visits blocks sequentially under a coded-bit budget C. The target retained count is $k_b^{\max} = \lceil 0.25 n_b \rceil$. If that record does not fit, the encoder reduces the retained count until a positive-count record fits, or skips the block if none does. For each admitted block, it retains the largest-magnitude coordinates at the admitted count and records their within-block positions. The selected values are quantized to INT8 using one FP32 scale shared by all selected coordinates from the same active layer. For an active layer ℓ,

$$s_\ell = \frac{\max_{j \in A_\ell} |\Delta_j|}{127}, \qquad q_j = \mathrm{clip}\left(\mathrm{round}\left(\frac{\Delta_j}{s_\ell}\right), -127, 127\right), \tag{2}$$

where A_ℓ denotes the selected coordinates in layer ℓ.

The payload contains the admitted block identifiers, local supports, INT8 values, and active-layer scales. Batch Normalization Through Time (BNTT) affine parameters and floating-point buffers

are carried as dense state within the same communication budget. We denote the resulting budget-constrained encode–decode operation by $\mathcal{C}_{\pi,C}$, where π is the block order and C is the bit budget.

3.2 LIF TEMPORAL EVIDENCE

Initial diagnostic replay. Before local training, each participating client performs a one-shot diagnostic replay of its first minibatch at the round-start model to extract timestep-wise LIF learning signals. The diagnostic replay uses the same SNN dynamics and surrogate backward rule as ordinary training. Unrolling the network over T timesteps gives a time-indexed occurrence $W_{\ell,t}$ of each shared layer weight W_ℓ. Backpropagating the final diagnostic cross-entropy loss $\mathcal{L}^{\mathrm{diag}}$, averaged over the minibatch, through this temporal graph yields the learning signal of block b at timestep t:

$$G_{b,t} = \widetilde{\nabla}_{W_{\ell(b),t}} \mathcal{L}^{\mathrm{diag}} \Big|_b, \tag{3}$$

where $\widetilde{\nabla}$ denotes surrogate-gradient backpropagation. The restriction $|_b$ extracts block b from the occurrence gradient.

Block magnitude and temporal evidence. For a vector v, define $\mathrm{RMS}(v) = \|v\|_2/\sqrt{\dim(v)}$. We express both the accumulated-update magnitude and the diagnostic magnitude relative to the RMS scale of their own layer. This makes the priority depend on relative within-layer strength rather than on raw layer scales:

$$\begin{aligned} r_b &= \mathrm{RMS}(\Delta_b), \qquad \widehat{r}_b = \frac{r_b}{\max\left\{\sqrt{|\mathcal{B}_\ell|^{-1}\sum_{c\in\mathcal{B}_\ell} r_c^2}, \varepsilon\right\}}, \\ q_{b,t} &= \mathrm{RMS}(G_{b,t}), \quad \widehat{q}_{b,t} = \frac{q_{b,t}}{\max\left\{\sqrt{|\mathcal{B}_\ell|^{-1}\sum_{c\in\mathcal{B}_\ell} q_{c,t}^2}, \varepsilon\right\}}, \qquad \ell = \ell(b), \end{aligned} \tag{4}$$

where $\varepsilon = 10^{-12}$ is used for numerical stability.

Temporal evidence weights the normalized diagnostic magnitude by the absolute cosine similarity between the diagnostic signal and the accumulated local update:

$$E_{b,t} = \widehat{q}_{b,t} \frac{|\langle G_{b,t}, \Delta_b\rangle|}{\max\{\|G_{b,t}\|_2, \varepsilon\}\max\{\|\Delta_b\|_2, \varepsilon\}}. \tag{5}$$

We then rescale the evidence by its RMS across all blocks at each timestep and clip each rescaled value before averaging. The resulting bounded multiplier modulates the layer-normalized block magnitude:

$$\begin{aligned} s_t &= \max\left\{\sqrt{\frac{1}{M}\sum_{b\in\mathcal{B}} E_{b,t}^2}, \varepsilon\right\}, \\ S_b &= \widehat{r}_b\left[1 + \frac{\lambda}{T}\sum_{t=1}^{T} \min\left\{\frac{E_{b,t}}{s_t}, c_{\mathrm{clip}}\right\}\right]. \end{aligned} \tag{6}$$

where λ controls the strength of temporal modulation and c_{clip} limits the influence of unusually large temporal evidence. We use $\lambda = 1$ and $c_{\mathrm{clip}} = 4$ throughout the experiments. Sorting S_b in descending order gives the block order π used by the codec.

3.3 SERVER AGGREGATION AND RESIDUAL-CORRECTED DOWNLINK

Let U_i^r denote client i's upload and $\widehat{\Delta}_i^r = \mathrm{decode}(U_i^r)$ its decoded weight update. The server averages the decoded updates:

$$\Delta_r = \frac{1}{B}\sum_{i\in\mathcal{C}_r} \widehat{\Delta}_i^r. \tag{7}$$

The aggregate is applied to the uncompressed server weights θ_r with unit step size.

For the downlink, the server adds the previous compression residual $\mathbf{R}_r$ to the current aggregate. Let $\pi_r^{\downarrow}$ denote the descending raw-RMS order of blocks in this corrected input. The same codec produces

Algorithm 1 FedBLADE: LIF-aware hierarchical encoding

Require: Shared blocks $\mathcal{B}$; caps $C_\uparrow, C_\downarrow$ including dense state; local training rule
Initialize $\mathbf{b}_0 = \theta_0$, $\mathbf{R}_0 = 0$, and common dense normalization state
for round $r = 0, \dots, R-1$ **do**
  Sample B clients uniformly without replacement to obtain $\mathcal{C}_r$
  for all $i \in \mathcal{C}_r$ **in parallel do**
    Load $\mathbf{b}_r$ and the synchronized dense state
    Replay the first scheduled minibatch once; retain $\{G_{b,t}\}_{t=1}^{T}$ from Eq. (3)
    Restore buffers, execution mode, and supported random states; take no diagnostic optimizer step
    Run ordinary local training and form $\Delta_i^r = w_i^r - \mathbf{b}_r$
    Compute S_b by Eqs. (4)–(6); stably sort to obtain π_i^r
    Encode in order π_i^r under $C_\uparrow$ using block-local Top-k, support coding, and layer-scaled INT8
    Form upload U_i^r from the encoded weight update and dense-state changes; keep no uplink residual
  end for
  Decode $\widehat{\Delta}_i^r = \mathrm{decode}(U_i^r)$ and average: $\Delta_r = B^{-1} \sum_{i \in \mathcal{C}_r} \widehat{\Delta}_i^r$
  Update $\theta_{r+1} = \theta_r + \Delta_r$; separately aggregate dense-state changes
  Form the corrected downlink input $\widetilde{\Delta}_r = \Delta_r + \mathbf{R}_r$
  Rank blocks by raw $\mathrm{RMS}(\widetilde{\Delta}_{r,b})$ and encode under $C_\downarrow$ with the same block codec
  Decode the downlink increment Q_r and set $\mathbf{R}_{r+1} = \widetilde{\Delta}_r - Q_r$
  Deliver Q_r and the synchronized dense state to all N clients; update $\mathbf{b}_{r+1} = \mathbf{b}_r + Q_r$
  Evaluate the server model θ_{r+1} when scheduled
end for

a decoded increment Q_r under the downlink cap $C_\downarrow$, and its reconstruction error becomes the next residual:

$$\begin{aligned} \widetilde{\Delta}_r &= \Delta_r + \mathbf{R}_r, \\ Q_r &= \mathcal{C}_{\pi_r^\downarrow, C_\downarrow}(\widetilde{\Delta}_r), \\ \mathbf{R}_{r+1} &= \widetilde{\Delta}_r - Q_r. \end{aligned} \tag{8}$$

The residual carries both omitted coordinates and quantization error into the next downlink input.

The server and clients therefore evolve with different increments:

$$\theta_{r+1} = \theta_r + \Delta_r, \qquad \mathbf{b}_{r+1} = \mathbf{b}_r + Q_r. \tag{9}$$

Here θ_r denotes the full server model, while $\mathbf{b}_r$ is the shared reconstruction used to initialize local training. Maintaining a common $\mathbf{b}_r$ requires every downlink increment Q_r to be delivered to all N registered clients, including those not selected for local training in round r. Algorithm 1 summarizes the complete procedure.

Encoding properties. The hierarchical encoder has three useful properties. First, the temporal term provides a bounded modulation of block magnitude:

$$\widehat{r}_b \le S_b \le \big(1 + \lambda c_{\mathrm{clip}}\big)\widehat{r}_b. \tag{10}$$

With $\lambda = 1$ and $c_{\mathrm{clip}} = 4$, the priority score lies between $\widehat{r}_b$ and $5\widehat{r}_b$, allowing temporal evidence to reorder blocks without unboundedly amplifying their magnitude scores. Second, for a fixed retained count k, selecting the k largest-magnitude coordinates within a block maximizes the retained squared update energy and equivalently minimizes the omission error before quantization. Finally, with $\mathbf{b}_0 = \theta_0$ and $\mathbf{R}_0 = 0$, the downlink recursion gives

$$\mathbf{R}_r = \theta_r - \mathbf{b}_r, \tag{11}$$

so the residual exactly tracks the discrepancy between the full server model and the compressed reconstruction shared by the clients. Proofs are provided in Appendix C.

4 Experiments

4.1 Setup

Datasets and training protocol. We evaluate FedBLADE on Fashion-MNIST, CIFAR-10, and CIFAR-100 using their standard training and test splits. To control statistical heterogeneity, the

training set of each dataset is partitioned across $N = 20$ clients using a class-wise Dirichlet distribution with $\alpha \in \{0.1, 0.3, 0.5\}$, where smaller α corresponds to stronger label skew. We use VGG5-BNTT with $T = 4$ timesteps for Fashion-MNIST, AlexNet-BNTT with $T = 4$ for CIFAR-10, and VGG9-BNTT with $T = 6$ for CIFAR-100. Unless otherwise specified, each experiment runs for 100 communication rounds. At every round, $B = 2$ clients are sampled uniformly without replacement, and each selected client performs five local epochs of SGD with batch size 128. The main budget sweep uses a learning rate of 0.003 for Fashion-MNIST and 0.05 for CIFAR-10 and CIFAR-100, with a unit server step size. For FedBLADE, the target within-block retention ratio is 25%, retained values are encoded in signed INT8, and one diagnostic replay is performed for each participating client and round. Experiments are conducted on Ascend 910B NPUs using AsPy, our in-house package for SNN execution and operator support. Additional architecture, optimizer, and extension-specific settings are given in Appendix D.

Communication accounting and evaluation. Communication budgets are specified per client and direction relative to dense model-state transmission. We enforce each cap using the bit lengths of the specified message fields, including values, support information, quantization scales, framing, and dense BNTT state. The reported communication measure is coded-field size rather than measured network traffic. Only the B selected clients upload, whereas each downlink increment is delivered to all N registered clients. The codec and recipient-summed delivery cost are specified in Appendices B.

Table 1: System-level comparison under Dirichlet label skew. FedBLADE uses a 5% per-client communication budget in each direction, while the external baselines retain their method-specific communication and participation protocols. Each entry reports Tail-10 test accuracy (%, over five seeds). The highest and second-highest means in each column are shown in bold and underlined.

Method	Fashion-MNIST			CIFAR-10			CIFAR-100		
	α=0.1	α=0.3	α=0.5	α=0.1	α=0.3	α=0.5	α=0.1	α=0.3	α=0.5
FedSNN (Venkatesha et al., 2021)	68.4±4.4	80.4±1.7	84.1±0.7	36.7±5.2	47.9±3.4	50.7±3.7	22.4±2.9	26.2±4.2	28.0±2.5
SFedCA (Zhan et al., 2025)	67.3±4.3	80.1±2.2	83.1±1.4	38.7±4.2	44.8±5.1	48.4±6.5	18.8±1.8	23.6±6.2	22.3±4.9
FedILSR (Wang et al., 2026)	61.4±10.5	79.0±3.0	81.5±2.4	29.5±1.8	41.2±3.2	45.6±3.3	18.3±0.7	25.3±1.7	22.7±4.3
FedLEC (Yu et al., 2024)	64.3±7.1	79.9±1.6	83.8±0.7	33.5±3.6	46.5±3.9	50.5±4.3	21.4±2.4	25.7±4.7	28.1±1.9
DivFL (Balakrishnan et al., 2022)	69.0±4.5	80.6±1.8	84.2±0.9	38.9±2.6	43.2±5.7	47.1±7.1	17.6±3.8	20.9±7.8	20.9±5.5
FedMoS (Wang et al., 2023b)	48.5±6.2	72.1±4.5	76.9±1.4	21.9±2.5	22.8±13.3	32.0±11.0	8.7±4.4	11.7±6.1	10.3±8.0
FedRS (Li & Zhan, 2021)	71.7±4.5	82.4±0.9	<u>84.4±0.6</u>	41.6±3.7	49.0±4.4	51.8±3.4	<u>25.6±3.6</u>	<u>27.6±4.6</u>	<u>28.6±2.4</u>
FedLC (Zhang et al., 2022)	<u>73.2±4.1</u>	**83.1±0.8**	**84.9±0.6**	<u>42.8±3.5</u>	**50.1±3.6**	<u>52.5±3.0</u>	**25.9±4.0**	**28.1±4.5**	**29.2±2.2**
FedBLADE *Ours*	**73.8±3.0**	<u>82.6±1.7</u>	84.3±1.2	**46.0±3.4**	**50.1±3.6**	**54.7±1.6**	19.8±2.0	26.9±5.2	26.4±2.9

4.2 Main Results

Table 1 shows that aggressive communication compression can remain competitive with established federated SNN methods. FedBLADE achieves higher mean accuracy than FedSNN across all Fashion-MNIST and CIFAR-10 settings despite using a 5% coded-bit cap in each direction, with the largest gap reaching 9.3 points on CIFAR-10 at $\alpha = 0.1$. The pattern changes on CIFAR-100, where FedLC and FedRS achieve higher mean accuracy. This contrast suggests that communication compression and label-skew correction address different bottlenecks: FedBLADE reduces the cost of transmitting updates, while methods such as FedLC and FedRS directly modify learning under statistical heterogeneity.

Table 2: Communication-compressor comparison. Column caps apply per client and direction to all methods except FLTS, which uses fixed $\kappa = 0.06$. Entries are Tail-10 accuracy (%, mean $\pm$ sample standard deviation over five seeds). Bold and underline mark the highest and second-highest reported means before rounding.

Method	Fashion-MNIST (1%)			CIFAR-10 (3%)			CIFAR-100 (5%)		
	α=0.1	α=0.3	α=0.5	α=0.1	α=0.3	α=0.5	α=0.1	α=0.3	α=0.5
Block-TOPK (Su et al., 2023)	**73.3±2.6**	**82.5±3.8**	**85.2±1.3**	<u>35.9±4.5</u>	<u>44.6±2.7</u>	<u>49.2±1.2</u>	<u>17.9±3.9</u>	<u>25.9±4.0</u>	<u>25.5±2.7</u>
Global-TopK (Stich et al., 2018)	68.8±4.2	<u>81.3±2.5</u>	81.6±5.7	30.3±3.4	39.6±4.7	45.6±2.6	14.9±3.1	24.2±4.4	24.8±2.5
STC (Sattler et al., 2019)	34.7±14.9	49.7±23.1	39.7±16.6	20.6±4.4	23.7±7.0	30.8±2.4	15.4±5.9	15.8±7.9	17.8±5.4
SSTC (Mora et al., 2022)	50.7±13.5	50.2±20.6	48.2±13.7	19.5±2.8	21.7±6.6	27.9±3.8	12.8±5.2	14.3±5.5	15.5±3.9
FLTS (Nguyen et al., 2024)	66.5±7.4	<u>81.3±1.9</u>	<u>83.6±1.1</u>	25.9±3.6	33.1±9.0	40.0±5.3	8.5±3.1	11.9±5.6	14.1±2.9
L-GreCo (Markov et al., 2024)	43.5±13.7	43.8±17.8	42.6±16.7	24.1±4.2	23.8±7.3	30.9±3.6	15.4±6.7	17.1±6.1	17.5±5.3
FedBLADE *Ours*	<u>70.3±3.6</u>	80.3±1.4	81.6±1.3	**44.9±3.9**	**49.5±3.4**	**53.9±2.2**	**19.8±2.0**	**26.9±5.2**	**26.4±2.9**

4.3 Comparison with Communication Compressors

Table 2 compares FedBLADE with compressors operating at different granularities. Global-TopK and STC perform coordinate-level sparsification, SSTC and Block-TOPK exploit structured parameter groups, and L-GreCo allocates sparsity across layers; FLTS is included as an SNN-specific reference. For the budget-adapted methods, all specified message fields must fit the per-client, per-direction caps shown in the table, including values, support information, codec metadata, and dense normalization state. Global-TopK is our budget-constrained INT8 implementation of largest-magnitude coordinate selection (Stich et al., 2018). FLTS uses a fixed sparsification ratio $\kappa = 0.06$ (Nguyen et al., 2024), corresponding to approximately 10.3% of dense FP32 communication under our value-and-index accounting, so its results are not cap-matched. Additional details are provided in Appendix D.4.

The results show that the benefit of FedBLADE depends strongly on the task. Block-TOPK leads on all three Fashion-MNIST settings by 2.2–3.6 points, showing that direct block-energy ranking is already highly effective for this task. The ordering reverses on CIFAR-10 and CIFAR-100, where FedBLADE achieves the highest mean accuracy in all six settings. Since both methods exploit block structure, these results suggest that the advantage of FedBLADE comes from how structured blocks are prioritized and encoded, not from block sparsity alone. They also show that the additional hierarchy is most useful when simple block-energy ranking leaves substantial accuracy loss.

Granularity matters as well. At $\alpha = 0.1$, Global-TopK trails FedBLADE by 14.6 points on CIFAR-10 and 4.9 points on CIFAR-100, indicating that preserving architecture-local structure can be critical under tight communication budgets. Taken together, the comparison supports hierarchical structured compression as a key design choice in the harder CIFAR settings, while Fashion-MNIST provides a useful counterexample where a simpler structured compressor is sufficient. Section 4.4 further examines which components of FedBLADE account for these differences.

4.4 Ablation Study

Table 3 reveals a clear hierarchy among the components of FedBLADE. The largest drops come from Global-TopK and No-Downlink-EF, while changes to the block ordering have smaller effects. No-INT8 is also 3.1 points below FedBLADE. Under a fixed bit cap, FP32 values consume more bits and reduce the number of coordinates that can be retained, illustrating the trade-off between numerical precision and transmitted support. Overall, the ablation shows that FedBLADE derives most of its robustness from structured representation and feedback, while the priority rule determines how the remaining limited capacity is allocated.

Table 3: Ablation comparisons on CIFAR-10 with $\alpha = 0.1$ and a 3% coded-bit cap per direction. Entries are Tail-10 accuracy (%, mean $\pm$ sample standard deviation over five seeds). Δ denotes the difference from the FedBLADE mean in percentage points.

Variant	Tail-10 accuracy	Δ
FedBLADE	**44.9±3.9**	–
UL-Shuffle	42.5±2.6	−2.4
Plain-RMS	41.8±2.8	−3.1
No-INT8	41.8±3.8	−3.1
Global-TopK	30.3±3.4	−14.6
No-Downlink-EF	30.1±5.1	−14.8

4.5 Communication Budgets

Figure 3 shows a clear budget-knee effect. The largest gains generally occur when moving out of the most constrained regime. On CIFAR-10, the largest accuracy gain occurs when increasing the cap from 1% to 3%, with diminishing improvements at higher budgets. Fashion-MNIST saturates even earlier, whereas CIFAR-100 continues to benefit up to approximately 10%.

One reason is that the fixed dense BNTT state consumes a larger fraction of very small budgets, leaving little capacity for sparse weight records. The later saturation of CIFAR-100 further shows that the useful communication regime depends on the model and task. This trend is also consistent with Proposition 4: once the cap is sufficient to admit all target block records, the decoded update becomes independent of the visitation order, and additional budget provides diminishing benefit to the priority mechanism.

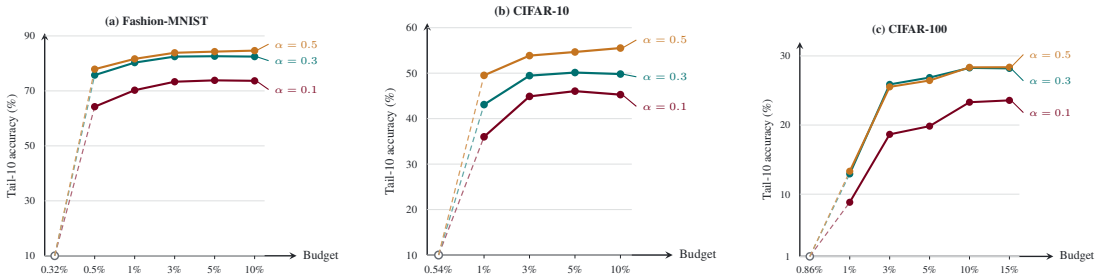


Figure 3: FedBLADE accuracy under different per-direction communication budgets. Each panel reports the five-seed mean Tail-10 test accuracy for $\alpha \in \{0.1, 0.3, 0.5\}$. Fashion-MNIST additionally includes the 0.5% budget. The hollow leftmost marker denotes the mandatory fixed-message budget paired with chance-level accuracy and serves only as a reference point; dashed segments are visual guides rather than measured results.

5 Conclusion

We presented FedBLADE, a LIF-aware hierarchical compressor that combines temporal block prioritization, compact within-block encoding, and residual-corrected downlink updates. The method makes block priority and representation cost explicit under a coded-bit budget. In the reported compressor comparisons, FedBLADE achieves the highest mean accuracy on CIFAR-10 and CIFAR-100, while Block-TOPK leads on Fashion-MNIST. The reported CIFAR-10 ablations show the largest mean decreases with global coordinate encoding or without downlink feedback. Future work includes combining compression with label-skew correction and measuring diagnostic and transport costs on edge devices.

AI USE STATEMENT

ChatGPT was used for manuscript drafting, inspection of the project code and recorded results. The empirical measurements originate from completed training runs. AI-assisted analysis recomputes or transcribes those records and does not supply new training outcomes. Responsibility for the manuscript and its claims rests with the authors.

ETHICS STATEMENT

The study uses established image-classification benchmarks and does not involve new collection of personal data. Keeping diagnostic traces local is a communication choice, not a formal privacy guarantee. We do not establish differential privacy or protection against update-inference attacks. The experiments do not measure device energy consumption, and no environmental or deployment-energy benefit is claimed.

REPRODUCIBILITY STATEMENT

Section 3 defines the priority score, codec, and state recursion. Appendix A details temporal-contribution extraction and diagnostic state restoration; Appendix B specifies message accounting and shared-state delivery; and Appendix C provides the encoding proofs. Training configurations, baseline adaptations, and ablation interventions are documented in Appendix D.

REFERENCES

Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. *Advances in neural information processing systems*, 30, 2017.

Ravikumar Balakrishnan, Tian Li, Tianyi Zhou, Nageen Himayat, Virginia Smith, and Jeff Bilmes. Diverse client selection for federated learning via submodular maximization. In *International conference on learning representations*, 2022.

Wei Fang, Zhaofei Yu, Yanqi Chen, Timothée Masquelier, Tiejun Huang, and Yonghong Tian. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2641–2651. IEEE, 2021.

Wei Fang, Yanqi Chen, Jianhao Ding, Zhaofei Yu, Timothée Masquelier, Ding Chen, Liwei Huang, Huihui Zhou, Guoqi Li, and Yonghong Tian. Spikingjelly: An open-source machine learning infrastructure platform for spike-based intelligence. *Science Advances*, 9(40):eadi1480, 2023.

Hong Huang, Weiming Zhuang, Chen Chen, and Lingjuan Lyu. Fedmef: Towards memory-efficient federated dynamic pruning. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 27538–27547. IEEE, 2024.

Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. Error feedback fixes signsgd and other gradient compression schemes. In *International conference on machine learning*, pp. 3252–3261. PMLR, 2019.

Mingjing Li, Huihui Zhou, Xiaofeng Xu, Zhiwei Zhong, Puli Quan, Xueke Zhu, Yanyu Lin, Wenjie Lin, Xiaosha Li, Dong Wang, et al. A highly energy-efficient multi-core neuromorphic architecture for training deep spiking neural networks. *Nature Communications*, 2026.

Xin-Chun Li and De-Chuan Zhan. Fedrs: Federated learning with restricted softmax for label distribution non-iid data. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pp. 995–1005, 2021.

Yujun Lin, Song Han, Huizi Mao, Yu Wang, and Bill Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SkhQHMW0W.

Wolfgang Maass. Networks of spiking neurons: the third generation of neural network models. *Neural networks*, 10(9):1659–1671, 1997.

Ilia Markov, Kaveh Alimohammadi, Elias Frantar, and Dan Alistarh. L-greco: Layerwise-adaptive gradient compression for efficient data-parallel deep learning. *Proceedings of machine learning and systems*, 6:312–324, 2024.

Alessio Mora, Luca Foschini, and Paolo Bellavista. Structured sparse ternary compression for convolutional layers in federated learning. In *2022 IEEE 95th Vehicular Technology Conference:(VTC2022-Spring)*, pp. 1–5. IEEE, 2022.

Manh V Nguyen, Liang Zhao, Bobin Deng, William Severa, Honghui Xu, and Shaoen Wu. The robustness of spiking neural networks in communication and its application towards network efficiency in federated learning. In *2024 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, pp. 1–7. IEEE, 2024.

Amirhossein Reisizadeh, Aryan Mokhtari, Hamed Hassani, Ali Jadbabaie, and Ramtin Pedarsani. Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. In *International conference on artificial intelligence and statistics*, pp. 2021–2031. PMLR, 2020.

Mengzhe Ruan, Guangfeng Yan, Yuanzhang Xiao, Linqi Song, and Weitao Xu. Adaptive top-k in sgd for communication-efficient distributed learning in multi-robot collaboration. *IEEE Journal of Selected Topics in Signal Processing*, 18(3):487–501, 2024. doi: 10.1109/JSTSP.2024.3381373.

Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Robust and communication-efficient federated learning from non-iid data. *IEEE transactions on neural networks and learning systems*, 31(9):3400–3413, 2019.

Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified sgd with memory. *Advances in neural information processing systems*, 31, 2018.

Junshen Su, Xijun Wang, Xiang Chen, and Ray-Guang Cheng. Joint sparsification and quantization for wireless federated learning under communication constraints. In *2023 IEEE 24th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 401–405. IEEE, 2023.

Ran Tao, Qiugang Zhan, Shantian Yang, Xiurui Xie, Qi Tian, and Guisong Liu. Sfedhifi: Fire rate-based heterogeneous information fusion for spiking federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 25878–25886, 2026.

Shubham Vaishnav, Sarit Khirirat, and Sindri Magnússon. Communication-adaptive-gradient sparsification for federated learning with error compensation. *IEEE Internet of Things Journal*, 12(2): 1137–1152, 2025. doi: 10.1109/JIOT.2024.3490855.

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

Jiarui Wang, Guoping Tan, Wei Cao, Weihua Cai, and Yue Yin. An energy-efficient spike-driven wireless federated learning framework for non-iid data. *Computer Networks*, pp. 112447, 2026.

Jue Wang, Yucheng Lu, Binhang Yuan, Beidi Chen, Percy Liang, Christopher De Sa, Christopher Re, and Ce Zhang. Cocktailsgd: Fine-tuning foundation models over 500mbps networks. In *International Conference on Machine Learning*, pp. 36058–36076. PMLR, 2023a.

Xiong Wang, Yuxin Chen, Yuqing Li, Xiaofei Liao, Hai Jin, and Bo Li. Fedmos: Taming client drift in federated learning with double momentum and adaptive selection. In *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications*, pp. 1–10, 2023b. doi: 10.1109/INFOCOM53939.2023.10228957.

Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 12:331, 2018.

Weiying Xie, Haowei Li, Jitao Ma, Yunsong Li, Jie Lei, Donglai Liu, and Leyuan Fang. Jointsq: Joint sparsification-quantization for distributed learning. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5778–5787. IEEE, 2024a.

Xiurui Xie, Jingxuan Feng, Guisong Liu, Qiugang Zhan, Zhetong Liu, and Malu Zhang. Federated learning for spiking neural networks by hint-layer knowledge distillation. *Applied Soft Computing*, 163:111901, 2024b.

Shuangqing Xu, Yifeng Zheng, and Zhongyun Hua. Harnessing sparsification in federated learning: A secure, efficient, and differentially private realization. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2354–2368, 2025.

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

Di Yu, Xin Du, Linshan Jiang, Huijing Zhang, and Shuiguang Deng. Exploiting label skewness for spiking neural networks in federated learning. *arXiv preprint arXiv:2412.17305*, 2024.

Qiugang Zhan, Jinbo Cao, Xiurui Xie, Huajin Tang, Malu Zhang, Shantian Yang, and Guisong Liu. Sfedca: Credit assignment-based active client selection strategy for spiking federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2025.

Chuanting Zhang, Haixia Zhang, Shuping Dang, Basem Shihada, and Mohamed-Slim Alouini. Gradient compression and correlation driven federated learning for wireless traffic prediction. *IEEE Transactions on Cognitive Communications and Networking*, 11(4):2246–2258, 2025. doi: 10.1109/TCCN.2024.3524183.

Jie Zhang, Zhiqi Li, Bo Li, Jianghe Xu, Shuang Wu, Shouhong Ding, and Chao Wu. Federated learning with label distribution skew via logits calibration. In *International conference on machine learning*, pp. 26311–26329. PMLR, 2022.

A NOTATION AND DIAGNOSTIC COMPUTATION

A.1 STATES AND INDICES

The vectors θ_r, $\mathbf{b}_r$, and $\mathbf{R}_r$ refer only to the D compressible convolutional and linear weights. Dense normalization parameters and floating-point buffers are synchronized separately. Table 4 collects the symbols used below.

Table 4: Notation used throughout the method and appendices.

Symbol	Meaning
r, i, t, ℓ, b	Communication round, client, timestep, layer, and block indices.
$N, B, \mathcal{C}_r$	Number of registered clients, participants per round, and the participating set.
$\theta_r, \mathbf{b}_r, \mathbf{R}_r$	Full server weights, shared client reconstruction, and downlink compression residual.
$\Delta_i^r, \Delta_r, \widetilde{\Delta}_r, Q_r$	Client update, aggregated decoded update, residual-corrected downlink input, and decoded downlink increment.
$\mathcal{B}, \mathcal{B}_\ell, M, n_b$	Block set, blocks in layer ℓ, total number of blocks, and size of block b.
$G_{b,t}, E_{b,t}, s_t$	Timestep-wise diagnostic learning signal, temporal evidence, and its timestep-specific normalization.
$r_b, \widehat{r}_b, S_b, \pi$	Raw block RMS, layer-normalized RMS, final priority score, and resulting block visitation order.

A scalar block index b is distinct from the vector $\mathbf{b}_r$. Similarly, scalar evidence $E_{b,t}$ is distinct from residual $\mathbf{R}_r$. The layer quantization scale s_ℓ in Eq. (2) and evidence scale s_t in Eq. (6) have different roles.

A.2 RECOVERING TEMPORAL CONTRIBUTIONS WITHOUT A TRAINING UPDATE

To recover the signals in Eq. (3), we separate the contributions from the time-indexed occurrences of each shared weight. For a linear layer, let $A_{\ell,t}$ contain the presynaptic activations and let $J_{\ell,t}$ contain the gradient of the final diagnostic loss with respect to the linear outputs, with examples stored as rows. The occurrence contribution is

$$G_{\ell,t} = J_{\ell,t}^{\mathsf{T}} A_{\ell,t}, \qquad \widetilde{\nabla}_{W_\ell} \mathcal{L}^{\mathrm{diag}} \Big|_b = \sum_{t=1}^{T} G_{b,t}. \tag{12}$$

For convolutions, the corresponding convolution-weight backward operation sums over examples and spatial positions. The loss reduction is already contained in $J_{\ell,t}$, so there is no further division by minibatch size. Equation (12) concerns one diagnostic graph; the multi-epoch endpoint update Δ_i^r is not assumed proportional to this gradient.

The replay uses the first scheduled minibatch before local SGD. It snapshots and restores normalization buffers, training mode, and supported CPU/accelerator random states, and performs no optimizer step. The diagnostic has its own seeded Poisson realization; restoring randomness does not mean its spikes coincide with those in the subsequent training pass. The diagnostic path uses packed layer operations with an eager surrogate recurrence, whereas ordinary NPU training uses the AsPy execution backend.

Zero vectors produce zero cosine evidence because their dot product is zero. All norm denominators use $\varepsilon = 10^{-12}$. Block RMS normalization is computed within each layer, while s_t is computed over all M blocks. Scores are averaged across timesteps once and sorted stably, with ties resolved by the fixed block order. The absolute cosine is a compatibility measure; it does not distinguish aligned from anti-aligned diagnostic contributions.

A.3 DIAGNOSTIC COST

For a client with K_i minibatches per epoch and E local epochs, the replay adds one diagnostic forward/backward pair to EK_i ordinary pairs. The diagnostic computation retains $O(TD)$ temporal gradient values and derives $O(TM)$ block statistics. Ranking the M blocks takes $O(M \log M)$ time. The evidence remains local, so it adds computation and memory but no evidence payload.

Algorithm 2 Budgeted block-local encoder used on either link

Require: Update v, order π, cap C, reservation H, shared block table
$L \leftarrow H + h_0$; $\mathcal{A} \leftarrow \varnothing$; $\mathcal{J} \leftarrow \varnothing$
for b in order π **do**
  for $k = \lceil 0.25 n_b \rceil, \ldots, 1$ **do**
    $h \leftarrow h_b(k) + 32\mathbf{1}\{\ell(b) \notin \mathcal{J}\}$
    if $L + h \le C$ **then**
      Set $k_b \leftarrow k$; add b to $\mathcal{A}$ and $\ell(b)$ to $\mathcal{J}$; $L \leftarrow L + h$
      break the inner loop
    end if
  end for
end for
Reject the cap if no positive-count block was admitted
for all $b \in \mathcal{A}$ **do**
  Select the k_b largest $|v_j|$, breaking ties by local index
  Encode the cheaper local support representation
end for
for all $\ell \in \mathcal{J}$ **do**
  Compute the scale from all selected values in ℓ and quantize by Eq. (2)
end for
Arrange fields canonically and include the reserved dense state
return the payload, decoded sparse vector, and accounted bits L

B BUDGETED ENCODING AND SHARED-STATE DELIVERY

B.1 WHAT IS COUNTED IN A MESSAGE

The dense reference F is the size of the stored model state at its declared element widths. For a budget fraction ρ, the cap is $C = \lfloor \rho F \rfloor$. Before selecting sparse weights, the encoder reserves H bits for BNTT affine parameters and floating-point buffers. The block table, tensor shapes, and layer order are common side information.

For M_ℓ blocks in layer ℓ, define index and inclusive-count widths

$$d(n) = \max\{1, \lceil \log_2 n \rceil\}, \qquad c(n) = \max\{1, \lceil \log_2 (n+1) \rceil\}.$$

Every layer contributes a block-count field, including inactive layers. A record for k retained coordinates in block b contains a layer-local block identifier, a support-mode bit, a coordinate count, local support, and encoded values:

$$\begin{aligned} h_0 &= \sum_{\ell} c(M_\ell), \\ h_b(k) &= d(M_{\ell(b)}) + 1 + c(n_b) + \min\{n_b, k d(n_b)\} + 8k, \\ L &= H + h_0 + \sum_{b \in \mathcal{A}} h_b(k_b) + 32|\mathcal{J}| \le C. \end{aligned} \tag{13}$$

Here $\mathcal{A}$ is the admitted block set and $\mathcal{J}$ its active layers. The local support is either an n_b-bit bitmap or a list of k local indices; ties favor the bitmap. Each active layer pays for one FP32 quantization scale, not one scale per block. For No-INT8, the value term becomes $32k$ and the scale term vanishes.

Relative-position coding means positions relative to the beginning of a block. Records are arranged by layer and local block identifier; bitmap values follow ascending support positions. Entries outside the transmitted support reconstruct to zero. For example, three values from a nine-weight block in a layer of 1,024 blocks cost $10 + 1 + 4 + 9 + 24 = 48$ record bits, before layer-scale and framing charges.

B.2 CODED BITS AND TRANSPORT BYTES

The implementation returns encoded values, support descriptors, and the coded-field bit count in Eq. (13). Communication results use this bit count rather than Python-object size or measured network

traffic. Byte alignment, transport headers, initialization, retransmissions, and reconnecting-client catch-up are outside the reported message accounting.

For the CIFAR-10 model, $F = 207{,}674{,}880$ bits and $H = 1{,}114{,}112$ bits. A 3% cap leaves $6{,}230{,}246 - H = 5{,}116{,}134$ bits for sparse records, scales, and their framing. The mandatory dense fractions are approximately 0.322%, 0.536%, and 0.858% for Fashion-MNIST, CIFAR-10, and CIFAR-100. These fractions exclude sparse framing and do not by themselves represent feasible nonzero-update training points. Figure 3 pairs them with chance accuracy only as reference markers.

B.3 Delivery required by the shared reconstruction

Equation (9) assumes a common initial state and delivery of every downlink increment to all registered clients, even when only B clients participate in training. This is the shared-base delivery contract (Protocol A). A client that misses increments must receive the missing information before it can use $\mathbf{b}_r$; obtaining the current base is not a free operation.

Let $L_{i,r}^{\uparrow}$ and $L_r^{\downarrow}$ include their dense reservations. Under recipient-summed accounting,

$$L_{\mathrm{round},r} = \sum_{i \in \mathcal{C}_r} L_{i,r}^{\uparrow} + N L_r^{\downarrow} \le (B+N)\lfloor \rho F \rfloor. \tag{14}$$

C Encoding properties and proofs

We establish properties of the priority rule, support selection, budget admission, and state recursion. They explain the mechanisms summarized at the end of Section 3.

C.1 Bounded temporal modulation

Proposition 1 (Influence on block priority). *For finite inputs, $\lambda \ge 0$, and $c_{\mathrm{clip}} \ge 0$, the score in Eq. (6) satisfies*

$$\widehat{r}_b \le S_b \le (1 + \lambda c_{\mathrm{clip}})\widehat{r}_b.$$

In particular, if $\widehat{r}_b > (1+\lambda c_{\mathrm{clip}})\widehat{r}_c$, block c cannot outrank block b.

Proof. All factors in $E_{b,t}$ are nonnegative. Each clipped ratio belongs to $[0, c_{\mathrm{clip}}]$, as does their mean. Multiplication by the nonnegative $\widehat{r}_b$ gives the bounds. The second claim follows from $S_b \ge \widehat{r}_b > (1+\lambda c_{\mathrm{clip}})\widehat{r}_c \ge S_c$. □

Setting $\lambda = 0$ recovers layer-normalized RMS ordering. It does not recover the Plain-RMS control, which ranks by unnormalized r_b. This distinction prevents attributing their complete difference to temporal evidence alone. The score can distinguish diagnostic sequences with identical accumulated updates and identical time-summed gradients. Consider two length-two blocks in one layer, both with $\Delta_b = (1, 0)$. Let the first block's two occurrence contributions be $(1, 0)$ and $(-1, 0)$, and the second block's be $(0, 1)$ and $(0, -1)$. Both time sums are zero and all diagnostic magnitudes are equal. Nevertheless, their evidence sequences are $(1, 1)$ and $(0, 0)$, giving $S_1 = 1 + \sqrt{2}$ and $S_2 = 1$ for $\lambda = 1$, $c_{\mathrm{clip}} = 4$.

C.2 Support selection and reconstruction error

Proposition 2 (Optimal support at a fixed retained count). *For a fixed block vector v_b and count k, retaining its k largest absolute coordinates minimizes squared omission error over all supports of size k.*

Proof. For support A of size k,

$$\|v_b - (v_b)_A\|_2^2 = \|v_b\|_2^2 - \sum_{j \in A} v_j^2.$$

The first term is fixed, so the error is minimized by maximizing the retained sum of squares. Exchanging a smaller retained magnitude with a larger omitted magnitude cannot decrease that sum; the Top-k support is therefore optimal. □

Proposition 3 (Omission and quantization are separate errors). *Let A be the selected support and A_ℓ its part in active layer ℓ. With exact-arithmetic scales $s_\ell = a_\ell/127$, where $a_\ell = \max_{j\in A_\ell} |v_j|$, nearest-integer quantization satisfies*

$$\|v - \widehat{v}\|_2^2 = \sum_{j\notin A} v_j^2 + \sum_{\ell} \sum_{j\in A_\ell} (v_j - s_\ell q_j)^2 \le \sum_{j\notin A} v_j^2 + \sum_{\ell} \frac{|A_\ell| a_\ell^2}{4\cdot 127^2}. \tag{15}$$

Proof. Omission and quantization affect disjoint coordinates, giving the equality. The chosen scale ensures $|v_j/s_\ell| \le 127$, and rounding introduces at most $s_\ell/2$ error per selected coordinate. Summing their squared errors gives the bound. The all-zero case is exact. □

The bound isolates the mathematical quantizer; finite-precision scale storage and arithmetic introduce additional rounding. Under a common cap, No-INT8 changes the admissible support as well as eliminating value quantization, so its accuracy difference is not a quantization-only effect.

C.3 Budget feasibility and representation saturation

Proposition 4 (Feasible admission and a sufficient order-independent cap). *If Algorithm 2 returns a payload, its accounted size does not exceed C. Let $k_b^{\max} = \lceil 0.25 n_b \rceil$ denote the target retained count for block b, and define*

$$C_{\text{all}} = H + h_0 + \sum_{b\in\mathcal{B}} h_b(k_b^{\max}) + 32|\{\ell(b) : b \in \mathcal{B}\}|.$$

For $C \ge C_{\text{all}}$, every block is admitted with its target count, and the decoded payload is independent of visitation order, given fixed support tie-breaking.

Proof. Each admission is made only after testing its entire incremental cost, including a newly opened layer's scale. Hence the running ledger remains at most C. If $C \ge C_{\text{all}}$, the running cost plus the next block's full target cost never exceeds the cost of encoding all target records. No backoff is required. The selected support and each layer's scale are then functions only of v and the target counts, and canonical serialization removes order dependence. □

C.4 Downlink residual and server–client consistency

Proposition 5 (Residual identity). *Under Eqs. (8)–(9), with $\mathbf{b}_0 = \theta_0$ and $\mathbf{R}_0 = 0$,*

$$\mathbf{R}_r = \theta_r - \mathbf{b}_r, \qquad \sum_{r=0}^{R-1} Q_r = \sum_{r=0}^{R-1} \Delta_r - \mathbf{R}_R. \tag{16}$$

Proof. The identity holds initially. Assuming it at round r,

$$\theta_{r+1} - \mathbf{b}_{r+1} = (\theta_r - \mathbf{b}_r) + \Delta_r - Q_r = \mathbf{R}_r + \Delta_r - Q_r = \mathbf{R}_{r+1}.$$

Induction proves the first equality. Summing $Q_r = \Delta_r + \mathbf{R}_r - \mathbf{R}_{r+1}$ proves the second. □

The residual tracks the compression discrepancy for the actual decoded uplink sequence. It does not restore information already omitted on the uplink or bound the final residual without further assumptions. The stated identity holds in exact arithmetic; an implementation check should allow ordinary floating-point roundoff.

D Experimental protocols and comparison definitions

D.1 Training configuration and evaluation

Training uses the standard training/test splits with class-wise Dirichlet label partitions. FedBLADE has $N = 20$ clients, $B = 2$ uniformly selected participants per round, five local epochs, batch size 128, and 100 rounds. SGD uses no momentum or weight decay, and the server step size is one.

Table 5: Task-specific FedBLADE settings and per-direction communication budgets.

Dataset	Model	T	Learning rate	Budgets per direction
Fashion-MNIST	VGG5-BNTT	4	0.003	0.5/1/3/5/10%
CIFAR-10	AlexNet-BNTT	4	0.05	1/3/5/10%
CIFAR-100	VGG9-BNTT	6	0.05	1/3/5/10/15%

Table 6: Baseline mechanisms as used in the system-level comparison.

Method	Evaluated adaptation and participation
FedSNN (Venkatesha et al., 2021)	Dense reference; select two clients before local training.
SFedCA (Zhan et al., 2025)	Train candidates, then select two using spike-activity credit; candidate computation exceeds that of two-client training.
FedILSR (Wang et al., 2026)	Probe candidate spike activity on a local-data subset, then train selected clients; the port does not reproduce wireless power/time allocation.
FedLEC (Yu et al., 2024)	Select before training; cross-entropy plus missing-label distillation from a frozen round-start teacher, with weight 0.1. Label-prior calibration is disabled.
DivFL (Balakrishnan et al., 2022)	Select by update diversity after obtaining candidate updates; this implementation does not reproduce the original cached-update communication savings.
FedMoS (Wang et al., 2023b)	Double-momentum/drift-control port with adaptive selection; sampling with replacement can yield fewer than two distinct clients.
FedRS (Li & Zhan, 2021)	Select before training; restrict local prediction to observed classes through the implemented missing-class mask.
FedLC (Zhang et al., 2022)	Select before training; class-count logit calibration with unit calibration strength and masking of missing classes.

Table 5 gives the task-specific settings. The codec targets 25% within each block, uses INT8 values and one FP32 scale per active layer, and sets $\lambda = 1$ and $c_{\text{clip}} = 4$. Uplink error feedback is disabled; downlink feedback is enabled.

The 0.5% Fashion-MNIST and 15% CIFAR-100 points in Figure 3 are additional budget settings evaluated with the same task-specific training configuration.

D.2 System-level baselines

Table 1 compares evaluated implementations, not a common compression intervention. The contextual baseline pipeline generally draws ten candidate clients and selects two; candidate computation and communication depend on the method. Table 6 states the relevant adaptations. All methods use the dataset/model pairing in Table 5, but these protocol differences preclude interpreting the table as equal total communication or equal compute.

D.3 Baseline hyperparameters

All baselines use the dataset/model pairing and common training settings reported in Table 5, unless otherwise stated. The table below lists only method-specific hyperparameters and implementation choices that differ from the common protocol.

D.4 Communication-compressor adapters

All methods in Table 2 except FLTS use the stated coded-bit cap per client and direction, including dense normalization state. FLTS retains its fixed sparsification ratio. The budget-adapted methods share the message-accounting convention, while their optimizer, residual, and execution settings follow the adaptations described below.

Table 7: Method-specific hyperparameters and implementation choices used in the reported comparisons.

(a) Federated-learning and SNN baselines

Method	Method-specific hyperparameters and settings
FedSNN (Venkatesha et al., 2021)	Dense model exchange with the standard cross-entropy local objective and uniform aggregation.
SFedCA (Zhan et al., 2025)	Client credit is computed from the change in firing activity before and after local training; the two activity evaluations use the same Poisson seed to reduce sampling variation.
FedILSR (Wang et al., 2026)	Probe fraction $= 0.2$ of local data; numerical floor 10^{-12}; population standard deviation is used when normalizing candidate activity scores.
FedLEC (Yu et al., 2024)	Distillation weight $= 0.1$; temperature $= 1$; frozen round-start teacher; cross-entropy plus conditional KL over missing classes; label-prior calibration is disabled.
DivFL (Balakrishnan et al., 2022)	Pairwise update distance uses the ℓ_2 norm; representative updates are chosen by facility-location greedy selection; candidate updates are refreshed every round.
FedMoS (Wang et al., 2023b)	Local correction coefficient $a = 0.1$; server momentum $\beta = 0.8$; proximal coefficient $\mu = 10^{-3}$; adaptive sampling follows the implemented replacement-sampling rule.
FedRS (Li & Zhan, 2021)	Uses the complete local label histogram; logits corresponding to classes absent from the local client are hard masked in the evaluated implementation.
FedLC (Zhang et al., 2022)	Calibration strength $\tau = 1$ with class-count correction $z_c - \tau n_{i,c}^{-1/4}$; locally absent classes are masked.

(b) Communication-compression baselines

Method	Method-specific hyperparameters and settings
STC (Sattler et al., 2019)	Tensor-wise magnitude Top-k selection; retained values are ternarized to $\{-\mu, 0, +\mu\}$; one 32-bit scale μ is transmitted per tensor; supports are Golomb coded; error feedback is applied on both links.
SSTC (Mora et al., 2022)	Kernel preselection fraction $= 0.125$; convolutional kernels are ranked by mean absolute update magnitude before sparse ternary compression; bidirectional residual accumulation is enabled in the evaluated adapter.
Block-TOPK (Su et al., 2023)	Block length $l = 128$; blocks are ranked by squared ℓ_2 norm; quantization precision is searched over 2–32 bits; retained values use stochastic uniform quantization; client residual decay is $0.9^{\Delta r}$.
Global-TopK (Stich et al., 2018)	Global largest-magnitude coordinate selection; retained values use INT8 with one FP32 scale per active layer; global coordinate identifiers encode the support; uplink error feedback is disabled.
FLTS (Nguyen et al., 2024)	Fixed sparsification ratio $\kappa = 0.06$ on both uplink and downlink; coordinates are ranked by model-update magnitude; transmitted values remain FP32; no error feedback is used.
L-GreCo (Markov et al., 2024)	Layer-wise allocation over Block-TOPK candidates with block length 128; candidate allocation multipliers are $\{0, \frac{1}{16}, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1\}$; the objective uses squared ℓ_2 compression error and dynamic programming with 1000 error bins.

Coordinate and kernel sparsification. STC (Sattler et al., 2019) selects coordinates by magnitude, ternarizes retained values, and accounts for compact support coding. SSTC (Mora et al., 2022) first selects convolutional kernels, using the configured fraction 0.125, and applies sparse ternary compression to that subset. The bidirectional adapters include residual accumulation; enabling it for SSTC is an adaptation rather than a paper-exact reproduction of the reference implementation.

Block and layer allocation. Block-TOPK (Su et al., 2023) uses contiguous fixed-length blocks, with configured length 128, and jointly chooses retained blocks and value precision under its cap. The original uplink-oriented mechanism is also applied to the downlink in this comparison. L-GreCo (Markov et al., 2024) denotes our layer-wise allocation adapter using a Block-TOPK-based candidate family, not the original data-parallel training system.

Global-TopK and FLTS. Global-TopK uses the largest-magnitude coordinate operator defined by Stich et al. (2018), instantiated here with an INT8 payload and global support identifiers. It changes the uplink selection/representation while retaining the FedBLADE downlink, and is an internal comparator, not a reproduction of memory-SGD. FLTS (Nguyen et al., 2024) exchanges sparsified model values on both links with fixed $\kappa = 0.06$ in the available records. A fixed coordinate ratio does not establish the same coded-bit cap, so those accuracies are not inserted into the empty FLTS cells.

D.5 Exact ablation interventions

Table 8 summarizes how each control modifies FedBLADE. The comparisons use the same dataset, Dirichlet concentration, local-work setting, and coded-bit cap, with all remaining components following the reference configuration.

The replay schedule is retained for the active uplink controls, even when its evidence is not used for ranking. No-INT8 tests the value/support trade-off under a cap, not quantization at fixed support. Global-TopK changes support representation as well as selection granularity. Plain-RMS and the shuffle controls test complete ordering rules.

Table 8: Definitions of the controls used in Table 3.

Variant	Change relative to FedBLADE
Plain-RMS	Replace the uplink order by descending raw $r_b = \mathrm{RMS}(\Delta_b)$ across all blocks. This removes both temporal modulation and layer normalization, and is not $\lambda = 0$ in Eq. (6).
UL-Shuffle	Permute block identities among positions belonging to the same layer and block length. Preserve the reference sequence of layer/length slots, the local codec, and downlink feedback. No magnitude-window matching is used.
No-INT8	Encode retained values in FP32 on both links and remove quantization-scale fields. The same cap admits a different, usually smaller, support.
Global-TopK	Select uplink coordinates by global absolute magnitude and use INT8 values with global support identifiers; retain the reference block-local downlink and its residual.
No-Downlink-EF	Compress the current aggregate without adding the previous downlink residual. Retain uplink prioritization and the block-local value codec.