MEMBRANE-AWARE DYNAMIC SPARSE TRAINING FOR FEDERATED SPIKING NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Federated dynamic sparse training has emerged as a promising paradigm for distributed learning with reduced communication overhead. However, existing approaches inherit connection importance estimation criteria from artificial neural networks and overlook the membrane state of postsynaptic neurons, which determines spike emission in spiking neural networks (SNNs). This mismatch leads to inaccurate importance estimation and suboptimal sparse rewiring in federated SNNs. To address this challenge, we propose Membrane-Aware Dynamic Sparse Training (MADST), which introduces three mechanisms: (1) membrane-gated pruning that scores active connections by postsynaptic threshold proximity; (2) activity-guided growth that couples presynaptic spike activity with postsynaptic firing readiness; and (3) support-first server alignment that reconciles heterogeneous client masks under a fixed per-layer density budget. Experiments on CIFAR-10 and CIFAR-100 with Spiking ResNet-18 show that MADST attains the highest accuracy in 10 of the 12 matched-density sparse settings under IID and non-IID conditions, with accuracy gains of up to 1.23 percentage points over the strongest baseline.

1 INTRODUCTION

Federated learning (FL) enables collaborative model training without exchanging raw data(McMahan et al., 2017), while spiking neural networks (SNNs) offer event-driven computation for resource-constrained devices. However, federated SNN training remains costly because dense models require clients to optimize and communicate all parameters. Federated dynamic sparse training (DST) addresses this issue by maintaining sparse subnetworks during training. FedDST learns a shared sparse topology under a fixed density constraint, while FLASH further supports heterogeneous clients with different density budgets (Bibikar et al., 2022; Babakniya et al., 2022). Nevertheless, these methods rely on Artificial Neural Network (ANN)-derived importance criteria, such as weight magnitude, gradient magnitude, and Taylor sensitivity (Evci et al., 2020; Molchanov et al., 2019), which overlook the state-dependent dynamics of SNN neurons.

Unlike ANNs, SNN computation depends on membrane potential evolution: a neuron spikes only when its membrane potential crosses the firing threshold (Maass, 1997; Gerstner et al., 2014). Therefore, the effect of a synaptic connection is determined not only by its weight or gradient, but also by the membrane state of its postsynaptic neuron. Threshold proximity has been used to guide sparse rewiring within a single SNN (Chen et al., 2025), but the criteria used in federated sparse training remain weight- or gradient-derived, and surrogate gradients (Wu et al., 2019) leave threshold-related information entangled with the error signal rather than available directly for topology selection.

We further observe this limitation empirically in federated Spiking ResNet-18. Appendix D illustrates this on a representative layer: neurons with identical firing rates sit at substantially different distances from the threshold, so firing activity alone does not characterize a neuron's contribution potential. Table 3 quantifies the effect. They motivate the question this paper is organized around: under a fixed sparse budget, can the coincidence between a presynaptic spike and a postsynaptic neuron's sensitivity near threshold select more effective connections in a federated SNN?

Based on this insight, we propose Membrane-Aware Dynamic Sparse Training (MADST), a federated SNN framework that introduces membrane-aware topology optimization. MADST integrates membrane-gated pruning, activity-guided growth, and support-first server alignment to improve

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.

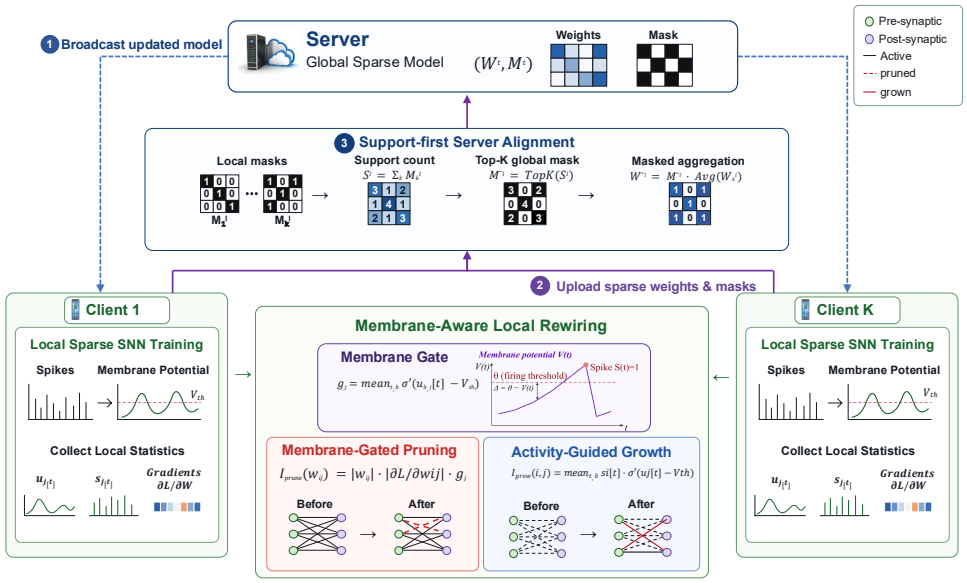


Figure 1: Overview of MADST. **Broadcast:** the server sends the updated global sparse model (W^t, M^t) to the participating clients. **Local training and upload:** each client trains a sparse SNN, collects spikes, membrane potentials, and gradients, applies membrane-gated pruning and activity-guided growth, and uploads sparse weights and masks. **Server alignment:** the server counts client support, selects a fixed-size global support with Top-K, and performs masked aggregation to produce the next global sparse model. Membrane and spike statistics remain local.

sparse connection selection while maintaining exact layer-wise density constraints. Membrane and spike statistics are only used locally and are never communicated. Figure 1 summarizes the framework.

We evaluate MADST on CIFAR-10 and CIFAR-100 under IID and non-IID partitions, including matched-density comparisons across three sparse density levels. Across these settings, MADST attains the highest accuracy in 10 of the 12 matched-density sparse comparisons.

Our contributions are summarized as follows:

- **A membrane-centric principle for sparse topology optimization in federated SNNs.** We identify a fundamental mismatch between conventional ANN-derived connection-importance criteria and the state-dependent dynamics of SNNs. We further show that firing activity alone cannot characterize a neuron's contribution potential, and establish the coincidence of presynaptic spiking with postsynaptic threshold sensitivity, rather than either signal alone, and score candidate connections by the zero-lag component of a connection's effect on its target.
- **A unified membrane-aware dynamic sparse training framework.** We propose MADST, which translates membrane dynamics into three coordinated topology-optimization mechanisms: membrane-gated pruning for preserving connections that are most likely to affect spike emission, activity-guided growth for activating synapses with high pre- and post-synaptic potential, and support-first server alignment for reconciling heterogeneous client masks under exact layer-wise density constraints. Support-first server alignment is what makes the other two usable in a federation: clients scoring connections from their own membrane statistics necessarily propose different supports, which must be reconciled without breaking the density budget. All membrane and spike statistics remain local, introducing no additional communication of neuronal states.
- **Accuracy–communication gains under heterogeneous federated settings.** Experiments with Spiking ResNet-18 on CIFAR-10 and CIFAR-100 show that MADST attains the high-

est accuracy in 10 of the 12 matched-density sparse settings spanning IID and non-IID partitions and three density regimes, with gains of up to 1.23 percentage points over the strongest competing baseline. It reaches this accuracy while communicating fewer model elements per round than dense federated learning, in the proportion the density budget prescribes, and adds nothing to that budget, since the membrane and spike statistics that drive its rewiring never leave the client.

2 Related Work

Dynamic Sparse Training.

Dynamic sparse training (DST) reduces computation and communication cost by maintaining and updating sparse subnetworks throughout training. Static criteria estimate connection importance before or during optimization, from single-batch sensitivity (Lee et al., 2019) to gradient flow (Wang et al., 2020) and data-free synaptic flow (Tanaka et al., 2020). DST instead rewires during training: SET prunes by magnitude and grows at random (Mocanu et al., 2018), and RigL selects growth by gradient (Evci et al., 2020). These methods were developed for ANNs and do not explicitly model postsynaptic membrane state.

Sparse SNN Training. Surrogate gradients enable end-to-end training through the non-differentiable spike function (Wu et al., 2019; Neftci et al., 2019), and SEW-ResNet supports deeper backbones (Fang et al., 2021). On these foundations, ESL-SNN rewires sparse connections during single-model training (Shen et al., 2023), structured methods prune channels by firing-related signals (Meng et al., 2023), Activity Pruning suppresses firing through adaptive thresholds (Bu et al., 2025), SLAMP allocates layer-wise sparsity before training (Wang et al., 2026), and SpQuant-SNN uses membrane-potential priors for spatial and channel pruning (Hasssan et al., 2024). Criticality-based pruning uses the derivative of the surrogate function as a low-cost proxy for how close a neuron sits to its firing threshold, and uses the resulting neuron score to drive the regeneration step of a prune-and-regenerate pipeline (Chen et al., 2025). Gradient rewiring reparameterizes connectivity so that pruning and regrowth follow the same gradient signal as weight learning (Chen et al., 2021), and dendritic-spine state transitions adapt sparse connectivity through stochastic state dynamics (Chen et al., 2022). These methods operate in a single-model setting and do not address the client-specific membrane states or the mask reconciliation required when sparse rewiring is distributed across federated clients.

Federated Sparse Training. Federated sparse training must reconcile both data heterogeneity and mask heterogeneity across clients. FedDST trains and exchanges shared sparse subnetworks under a fixed density (Bibikar et al., 2022), FLASH extends this to heterogeneous client resources with per-client density budgets (Babakniya et al., 2022), and FedSGC selects sparse updates using the congruity between local and global gradients (Tian et al., 2024). MADST addresses this federated mask-selection problem with an SNN-specific threshold-proximity signal. Our experiments compare against FLASH-FL, whose fixed per-layer density budget matches our setting most directly.

Federated Learning with SNNs. Prior work established federated SNN training for edge intelligence (Skatchkovsky et al., 2020; Venkatesha et al., 2021), hierarchical or asynchronous aggregation (Aouedi et al., 2023; Wang et al., 2023), and cross-scale fusion (Tao et al., 2026), focusing on training, aggregation or model heterogeneity. MADST instead addresses connection selection, using each client's membrane state to decide which connections stay active.

3 Method

In this section, we formulate the federated sparse SNN training problem and present MADST. We first introduce the sparse mask formulation, fixed layer-wise density constraints, and masked LIF dynamics. Based on this formulation, MADST performs membrane-aware sparse rewiring through three components: membrane-gated pruning, activity-guided growth, and support-first server alignment. Specifically, membrane-gated pruning augments the Taylor-based importance score with postsynaptic threshold proximity for connection selection, activity-guided growth regrows sparse connections by jointly considering presynaptic spike activity and postsynaptic firing readiness, and support-first server alignment reconciles heterogeneous client masks while preserving the target

density of each layer. Together, these components constitute MADST for efficient and adaptive federated sparse SNN training.

Algorithm 1 in Appendix A states the full federated procedure, and Appendix C reports the mechanistic, communication and energy analyses. Appendix E fixes the sparse-mask formulation, the fixed per-layer density constraint, and the masked LIF dynamics that the rest of this section builds on.

3.1 Membrane-Gated Pruning

In sparse federated SNN training, each client periodically prunes a fraction of its active connections and grows the same number of inactive ones to maintain a fixed density. Standard criteria based on weight magnitude or first-order loss sensitivity rank individual connections but do not explicitly control how sparse capacity is allocated across postsynaptic targets with different membrane states. We address this limitation with **membrane-gated pruning**, which augments the Taylor score with a target-level membrane gate. For a connection w_{ij}, removing the weight gives the first-order estimate (Molchanov et al., 2017; 2019)

$$\Delta\mathcal{L} \approx -\frac{\partial\mathcal{L}}{\partial w_{ij}} \cdot w_{ij}, \qquad |\Delta\mathcal{L}| \approx |w_{ij}| \cdot \left|\frac{\partial\mathcal{L}}{\partial w_{ij}}\right|. \tag{1}$$

We augment this connection-level score with a target-level measure of threshold proximity. To obtain this measure, we reuse the surrogate derivative σ' employed during backpropagation. For the ATan surrogate in Equation 27, σ' peaks at the threshold and decreases with absolute threshold distance. The fully connected gate is

$$g_j^{\text{fc}} = \frac{1}{TB}\sum_{t=1}^{T}\sum_{b=1}^{B}\sigma'(u_{b,j}[t] - V_{\text{th}}), \tag{2}$$

where B is the batch size and $u_{b,j}[t]$ is the pre-reset membrane potential. Averaging over samples and time steps yields a target-level summary of threshold proximity along the observed membrane trajectory. The same averaged surrogate derivative has been proposed as a neuron criticality score in single-model SNN pruning, where it selects which neurons to regenerate after pruning (Chen et al., 2025). We use the statistic at a different point in the rewiring loop: rather than scoring neurons for regeneration, g_j multiplies a connection-level sensitivity term during *pruning*, so it redistributes a fixed connection budget across surviving targets while leaving the ordering within each target untouched. Our growth rule conditions the same gate on presynaptic activity (Equation 8) rather than using it alone.

We combine the connection-level Taylor score with this target-level gate:

$$I_{\text{prune}}(w_{ij}) = |w_{ij}| \cdot \left|\frac{\partial\mathcal{L}}{\partial w_{ij}}\right| \cdot g_j^{\text{fc}}. \tag{3}$$

Multiplication implements a joint retention rule: a connection receives a high score only when it is both loss-sensitive and associated with a target whose observed membrane states have high average threshold proximity. The gate is detached from the optimization graph and used only during the discrete mask update.

For a convolutional layer with weight $W_{o,i,p,q}$ and postsynaptic membrane tensor $u_{b,o,h,w}[t]$, we use one gate per output channel:

$$g_o^{\text{conv}} = \frac{1}{TBH_oW_o}\sum_{t,b,h,w}\sigma'(u_{b,o,h,w}[t] - V_{\text{th}}). \tag{4}$$

Broadcasting g_o^{conv} to every kernel position (i,p,q) in output channel o gives

$$I_{\text{prune}}(W_{o,i,p,q}) = |W_{o,i,p,q}| \left|\frac{\partial\mathcal{L}}{\partial W_{o,i,p,q}}\right| g_o^{\text{conv}}. \tag{5}$$

Because incoming weights of the same fully connected neuron, or kernel elements of the same output channel, share one gate, the method preserves their within-target Taylor ordering while adjusting

retention priority across targets or channels. For each layer with weight tensor W and binary mask M, we evaluate all active positions, prune the lowest-scoring fraction π_t, and set the corresponding mask entries to zero.

What the membrane gate adds to the Taylor score. The gate g_j is built from the same surrogate derivative σ' that appears in backpropagation, but it enters the score differently, as expanding the chain rule for the masked LIF update makes explicit. Writing $\delta_{b,j}[t] = \partial\mathcal{L}/\partial s_{b,j}[t]$ for the error arriving at the spike output,

$$\frac{\partial \mathcal{L}}{\partial w_{ij}} = \frac{1}{\tau_m} \sum_{t=1}^{T} \sum_{b=1}^{B} \delta_{b,j}[t]\, \sigma'(u_{b,j}[t] - V_{\text{th}})\, s_{b,i}[t] + \mathcal{R}_{ij}, \tag{6}$$

where $\mathcal{R}_{ij}$ collects the contributions routed through the reset and leak paths of later steps. The gate enters multiplicatively, and its role is a specific one. Within a target g_j is constant, so it leaves every within-target comparison unchanged; across targets it applies a common bias, so it decides how the layer's fixed budget is divided between targets whose Taylor terms are comparable. That bias is computed from a statistic the Taylor term does not contain: g_j averages σ' non-negatively rather than summing signed errors, does not depend on the presynaptic index, and is label-independent, whereas $\delta_{b,j}[t]$ is client-specific under non-IID partitions. The claim we make is therefore about *allocation across targets*, not about restoring information the gradient discards.

3.2 Activity-Guided Growth

After pruning, each client activates the same number of inactive positions to restore the density of every layer. Unlike active connections, these candidates have no trained weight magnitude. Gradient-based regrowth methods such as RigL rank them using dense gradients (Evci et al., 2020); MADST instead reuses the spike and membrane traces already collected during sparse SNN training, avoiding dense-gradient evaluation over inactive positions.

The LIF update exposes the two neural states relevant to a newly grown connection. Holding the previous membrane state and other inputs fixed, perturbing a candidate weight changes the target potential by

$$\Delta u_j[t] = \frac{1}{\tau_m} s_i[t]\, \Delta w_{ij}. \tag{7}$$

A source spike carries input through the candidate connection, while the target gate measures the sensitivity of the postsynaptic membrane trajectory around the firing threshold.

For a fully connected layer, we score a candidate connection (i, j) by how often source neuron i fires at the same sample and time step at which target neuron j sits near its firing threshold:

$$I_{\text{grow}}(i,j) = \frac{1}{TB} \sum_{t=1}^{T} \sum_{b=1}^{B} s_{b,i}[t]\, \sigma'(u_{b,j}[t] - V_{\text{th}})\,. \tag{8}$$

The score measures the coupling between the two neural states rather than their marginals. A candidate perturbs the target through a strictly causal kernel. Unrolling the LIF recursion behind Equation 7 across steps, with the target's spike pattern held fixed,

$$\frac{\partial u_j[t]}{\partial w_{ij}} = \frac{1}{\tau_m} \sum_{\Delta \geq 0} \Big(1 - \frac{1}{\tau_m}\Big)^{\Delta} s_i[t-\Delta] \prod_{r=t-\Delta}^{t-1} \big(1 - s_j[r]\big),$$

so a source spike reaches the target's spike decision at its own step and, damped geometrically in the lag Δ, at later steps until the target fires. Equation 8 scores the zero-lag component of this kernel, the term that carries the full coefficient $1/\tau_m$ and that no reset can cancel. The lagged terms are the growth-side counterpart of $\mathcal{R}_{ij}$ in Equation 6 and are left out, so the rule prices a candidate by its immediate effect on the target's spike decision rather than by its full temporal sensitivity. The expansion describes the layer's own dynamics; the backbone applies batch normalization between the convolution and the LIF neuron, which rescales the perturbation. Collecting the traces into $S \in \{0,1\}^{TB \times n_{\text{in}}}$ and $G \in \mathbb{R}^{TB \times n_{\text{out}}}$, the score matrix is $\frac{1}{TB} S^\top G$, accumulated across minibatches

into a weight-shaped buffer so no trace outlives the step that produces it. Because the score is assembled from forward-pass traces alone, candidate positions are ranked without a gradient defined at them. The selected positions are activated by setting $M_{ij} = 1$, initialized at the scale used for the same layer, and optimized during subsequent local training.

For a convolutional layer the same coupling is resolved per kernel offset. A kernel coordinate (o, i, p, q) connects output position (h, w) to input position $(h+\delta p,\ w+\delta q)$, where $\delta p = p - \lfloor k/2 \rfloor$ for a stride-one layer with equal padding, so the score is the spatial cross-correlation between the presynaptic spike map of channel i and the gate map of channel o evaluated at that offset:

$$I_{\text{grow}}(o, i, p, q) = \frac{1}{TBH_oW_o} \sum_{t,b,h,w} \sigma'(u_{b,o,h,w}[t] - V_{\text{th}}) \times\ s_{b,i,\,h+\delta p,\,w+\delta q}[t]. \tag{9}$$

Scoring each kernel coordinate separately, rather than sharing one value across a channel pair, selects the spatial offset of a grown connection on evidence and removes the need for a tie-breaking rule. It reuses an existing operator: Equation 9 has the form of a convolutional weight gradient with the gate map in place of the output gradient, accumulated across minibatches into a weight-shaped buffer.

3.3 Support-First Server Alignment

Local rewiring produces different sparse supports because clients observe different data, membrane potentials, and spike traces. Here, the support of a mask denotes its active connection positions. Directly averaging binary masks would create fractional mask values and would not guarantee the target density. We therefore introduce **support-first server alignment**, which constructs a fixed-size global support using client agreement as the primary ranking key and aggregated weight magnitude as the secondary key, projecting onto exactly K_l coordinates so that the prescribed density holds by construction rather than in expectation.

Making agreement the *primary* key is what adapts this step to the spiking setting. A client's weight magnitudes are a product of its own membrane and firing statistics, so under non-IID data the same coordinate carries different magnitudes at different clients and a magnitude-led rule follows whichever client is most active. A vote count carries no such scale and is comparable across clients by construction, so this ordering anchors the global support to consensus and leaves magnitude to break ties, while projecting onto exactly K_l coordinates makes the prescribed density hold by construction rather than in expectation. Both ingredients have precedent: FedDST decides the server mask from weight magnitudes and client mask votes (Bibikar et al., 2022), and majority voting over client supports appeared earlier in sparse distributed SGD (Ozfatura et al., 2020).

For layer l, let $\mathcal{P}_l$ be the set of weight coordinates, M_k^l and W_k^l the sparse mask and weights uploaded by client $k \in S_t$, and n_k the number of local training examples. For every position, the server counts how many client masks contain it:

$$C^l = \sum_{k \in S_t} M_k^l. \tag{10}$$

We select exactly $K_l = \lfloor \rho_l |W_l| \rfloor$ positions for each layer. Following Bibikar et al. (2022), we define the mask-weighted average

$$\bar{W}^l_{\text{raw}} = \frac{\sum_{k \in S_t} n_k (W_k^l \odot M_k^l)}{\sum_{k \in S_t} n_k M_k^l + \epsilon}. \tag{11}$$

The final mask keeps the K_l positions with the largest lexicographic keys $(C_p^l, |\bar{W}^l_{\text{raw},p}|)$, comparing client support first and weight magnitude second:

$$\mathcal{A}_l = \text{TopK}^{\text{lex}}_{K_l}\Big(\big(C_p^l,\ |\bar{W}^l_{\text{raw},p}|\big)_{p \in \mathcal{P}_l} \Big),$$

$$\bar{M}_p^l = \mathbb{1}[p \in \mathcal{A}_l], \tag{12}$$

$$\bar{W}^l = \bar{M}^l \odot \bar{W}^l_{\text{raw}}. \tag{13}$$

where $\text{TopK}^{\text{lex}}_{K_l}$ returns coordinate indices and a fixed coordinate order resolves any remaining ties. Since every participating client uploads exactly K_l active positions, the union of their supports contains at least K_l coordinates. Equation 12 therefore selects only coordinates with positive client support. The numerator and denominator in Equation 11 are evaluated elementwise, and ϵ prevents division by zero at zero-support coordinates.

Algorithm 1 Federated training with MADST

Require: Client set $\mathcal{K} = \{1, \ldots, N\}$ with local data $\mathcal{D}_k$; target layer densities; rewiring schedule and ratios
Require: Communication rounds; local epochs
Ensure: Trained global sparse model
Initialize M^0 randomly at the target densities and initialize W^0 on the active positions.
for $t = 0, \ldots, R-1$ **do**
  Set $S_t \leftarrow \mathcal{K}$; broadcast (W^t, M^t).
  for all $k \in S_t$ in parallel **do**
    $(W_k, M_k) \leftarrow (W^t, M^t)$.
    Train the masked SNN locally; retain the gradients, membrane potentials, and spikes used for rewiring.
    if t belongs to the predefined rewiring schedule **then**
      Compute (g_k, r_k) from (u_k, s_k).
      for all layers l **do**
        $q_l \leftarrow \lfloor \pi_t \| M_k^l \|_0 \rfloor$.
        Compute I_{prune} on active positions with Equation 3.
        Set the q_l lowest-scoring active mask entries to zero.
        Compute I_{grow} on inactive positions with Equation 8.
        Activate the q_l highest-scoring positions and initialize their weights at the scale of layer l.
      end for
    end if
    Upload (W_k, M_k).
  end for
  for all layers l **do**
    $C^l \leftarrow \sum_{k \in S_t} M_k^l$.
    Compute $\bar{W}_{\text{raw}}^l$ with Equation 11.
    $K_l \leftarrow \lfloor \rho_l |W_l| \rfloor$.
    Compute $(\bar{M}^l, \bar{W}^l)$ with Equations 12 and 13.
  end for
  $(W^{t+1}, M^{t+1}) \leftarrow (\bar{W}, \bar{M})$.
end for
return (W^R, M^R).

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Datasets and models. We used the standard CIFAR-10 and CIFAR-100 image-classification datasets (Krizhevsky, 2009), each with 50K training images, and a Spiking ResNet-18 (Fang et al., 2021).

Federated setting. We simulated 10 clients with full participation in every round. CIFAR-10 used 500 communication rounds and CIFAR-100 used 800 rounds. IID data were split uniformly, while non-IID data used LDA with $\alpha = 0.5$. Each client used SGD with momentum 0.9, weight decay 5×10^{-4}, cosine learning-rate decay, 2 local epochs, and batch size 32. The SNN used $T = 10$ time steps, LIF parameters $\tau_m = 2.0$ and $V_{\text{th}} = 0.3$, hard reset to zero, and an ATan surrogate gradient with $\alpha = 2.0$. Convolutional and linear weight layers were sparse; normalization and other weight-free layers remained dense. Masks were initialized randomly at the target density, and each sparse layer preserved its density after rewiring. We report the best test accuracy observed over the communication rounds, applying the identical rule to every method. Appendix B lists the remaining optimization settings, the conventions used for the first convolution and the linear readout, the federated adaptation of ESL-SNN, and the definition of each ablation variant.

Baselines and evaluation protocol. We compare MADST with static sparse training, FLASH-FL, and ESL-SNN using Spiking ResNet-18 at densities of $0.25\times$, $0.50\times$ and $0.75\times$. We exclude the full-density budget from the comparison: at $\rho = 1$ we have $K_l = P_l$, so the mask is identically one, and after pruning the q_l lowest-scoring connections the only inactive coordinates are exactly those just pruned, which the growth step then restores. These baselines cover a fixed sparse topology, federated sparse training, and SNN-specific dynamic sparse training. The comparison uses the same data partitions, backbone, optimizer, learning-rate schedule, local epochs, and communication

Dataset	**Density**	Static	FLASH-FL	ESL-SNN	MADST
CIFAR-10	$0.25\times$	77.25	82.22	83.52	**83.79**
CIFAR-10	$0.50\times$	79.40	85.19	85.43	**85.65**
CIFAR-10	$0.75\times$	85.01	84.91	84.53	**85.58**
CIFAR-100	$0.25\times$	51.03	59.33	59.69	**59.77**
CIFAR-100	$0.50\times$	57.76	59.25	60.36	**61.02**
CIFAR-100	$0.75\times$	57.57	52.33	58.94	**59.58**
$1.00\times$ omitted: at $\rho = 1$ the mask is complete, so no method can rewire.					

Table 1: Best test accuracy (%) with Spiking ResNet-18 on IID splits.

Dataset	**Density**	Static	FLASH-FL	ESL-SNN	MADST
CIFAR-10	$0.25\times$	75.24	74.78	81.30	**82.08**
CIFAR-10	$0.50\times$	79.49	**82.33**	80.15	81.83
CIFAR-10	$0.75\times$	80.28	80.07	81.50	**82.61**
CIFAR-100	$0.25\times$	47.53	55.31	56.20	**56.60**
CIFAR-100	$0.50\times$	54.40	**59.47**	57.48	58.19
CIFAR-100	$0.75\times$	53.67	53.69	55.23	**56.46**
$1.00\times$ omitted: at $\rho = 1$ the mask is complete, so no method can rewire.					

Table 2: Best test accuracy (%) with Spiking ResNet-18 on non-IID splits with LDA $\alpha = 0.5$.

rounds for every method. The same best-test reporting rule is applied to every method. Throughout the paper, *MADST* denotes the complete framework comprising Equations 3, 8, and 12.

4.2 Main Results

Overall performance. MADST ranks first in 10 of the 12 sparse settings. Under non-IID data it ranks first in four of six comparisons, with its largest gains at $0.75\times$ density: 1.11 percentage points on CIFAR-10 and 1.23 percentage points on CIFAR-100. Averaged over the 12 sparse settings, MADST reaches 71.10%, compared with 70.36% for ESL-SNN, 69.07% for FLASH-FL, and 66.55% for static sparsity.

Rewiring schedule and support consensus. At $0.50\times$ under non-IID data, MADST initially leads FLASH-FL but does not recover after the late-stage reversal; client mask overlap rises rapidly, so support consensus forms before the shared topology improves. This indicates that a predetermined rewiring schedule keeps perturbing weights after useful topology evolution has stopped, even once π_t has decayed, and that the schedule should instead be gated on whether the support is still changing.

4.3 Component Ablation

We ablate components at $0.75\times$ density, where moderate sparsity preserves capacity while still requiring nontrivial topology selection. Each *w/o* variant replaces the named rule; *w/o (MGP+AGG)* applies both substitutions at once, removing the membrane and the presynaptic activity signal together, while retaining SFA.

As shown in Table 3, Full MADST is the strongest variant in all four settings. Replacing SFA is the largest effect on CIFAR-100 under both partitions and produces the largest average drop across the four settings, and replacing activity-guided growth produces the largest single drop anywhere in the table, 2.11 points on CIFAR-10 under non-IID data.

4.4 Communication and Energy

At a matched density every method transmits the same payload, and MADST adds nothing to it: its membrane and spike traces are consumed on the client and never uploaded. At $0.50\times$ density the ten clients together upload 55.86×10^6 model elements per round against 111.74×10^6 for dense federated training, a 50.0% reduction, and per-inference energy, estimated from one client's

Setting	Full	w/o (MGP+AGG)	w/o AGG	w/o MGP	w/o SFA
CIFAR-10 IID	**85.58**	85.39	85.37	85.32	85.28
		-0.19	-0.21	-0.26	-0.30
CIFAR-10 non-IID	**82.61**	82.23	80.50	81.84	82.13
		-0.38	$\mathbf{-2.11}$	-0.77	-0.48
CIFAR-100 IID	**59.58**	59.27	59.52	59.31	58.69
		-0.31	-0.06	-0.27	-0.89
CIFAR-100 non-IID	**56.46**	55.91	56.23	56.18	54.65
		-0.55	-0.23	-0.28	$\mathbf{-1.81}$

Table 3: Component ablation at $0.75\times$ density. Each setting gives best test accuracy (%) and, below it, the change relative to the full method when one rule is replaced: MGP by the plain Taylor score, AGG by uniform random growth, SFA by a magnitude-only Top-K_l, and MGP+AGG by both substitutions at once. The two largest effects are shown in bold.

converged firing rates at 45 nm (Horowitz, 2014), is 0.049–0.146 mJ across the three budgets against 2.555 mJ for an equivalent dense one-step ANN (Appendix C).

5 Conclusion

MADST answers the question of the introduction with three mechanisms: activity-guided growth scores candidates by the coincidence of a presynaptic spike and a postsynaptic neuron's sensitivity near threshold, membrane-gated pruning allocates the retained budget across targets, and support-first alignment reconciles the supports these local decisions produce at an exact density. The statistics it needs are already materialized by sparse SNN training and never leave the client, so the communication budget is unchanged. On CIFAR-10 and CIFAR-100 across IID and non-IID partitions and three density budgets, MADST attains the highest accuracy in 10 of 12 matched-density settings. Our ablation separates the three mechanisms: replacing growth costs 2.11 points on CIFAR-10 under non-IID data, the largest single drop in the table, and reconciliation produces the largest average drop across the four settings.

6 Limitations and Future Work

MADST assumes that clients share layer shapes and expose pre-reset membrane potentials during local training. Applying it to heterogeneous architectures would require an explicit coordinate mapping before client supports could be compared. Our evaluation is limited to CIFAR-10 and CIFAR-100, full client participation, and fixed per-layer densities. Future work should extend MADST to event-based datasets, larger-scale tasks, partial and asynchronous client participation, and clients with heterogeneous model widths or architectures. Other directions include adaptive rewiring schedules based on support convergence, layer-wise or client-specific density budgets, and analyses of the communication and indexing overhead introduced by sparse mask exchange. Finally, hardware measurements are needed to validate the operation-level energy estimates under realistic memory access, sparse decoding, device utilization, and privacy constraints.

AI use statement

We used a large language model for language polishing of the manuscript: improving grammar, wording, and readability of text written by the authors. It was not used for any task in the required-disclosure categories of the ICLR AI policy. In particular, the research idea, the method, its implementation, the experimental design, and every reported number are the authors' own, and no generative AI was used to develop the theoretical claims, write the proofs, generate data, implement the method, run experiments, or interpret results. The authors reviewed all AI-assisted text against the equations and experimental records, and take responsibility for the final content of this work.

REFERENCES

Ons Aouedi, Kandaraj Piamrat, and Mario Südholt. Hfedsnn: Efficient hierarchical federated learning using spiking neural networks. In *Proceedings of the International ACM Symposium on Mobility Management and Wireless Access (MobiWac)*, pp. 53–60, 2023. doi: 10.1145/3616390.3618288.

Sara Babakniya, Souvik Kundu, Saurav Prakash, Yue Niu, and Salman Avestimehr. Lottery aware sparsity hunting: Enabling federated learning on resource-limited edge. *arXiv preprint arXiv:2208.13092*, 2022.

Sameer Bibikar, Haris Vikalo, Zhangyang Wang, and Xiaohan Chen. Federated dynamic sparse training: Computing less, communicating less, yet learning better. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 6080–6088, 2022. doi: 10.1609/aaai.v36i6.20555.

Tong Bu, Xinyu Shi, and Zhaofei Yu. Activity pruning for efficient spiking neural networks. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Shuo Chen, Boxiao Liu, Haihang Liu, and Peng You. Brain-inspired efficient pruning: Exploiting criticality in spiking neural networks. *Concurrency and Computation: Practice and Experience*, 37(27-28), 2025.

Yanqi Chen, Zhaofei Yu, Wei Fang, Tiejun Huang, and Yonghong Tian. Pruning of deep spiking neural networks through gradient rewiring. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1713–1721, 2021.

Yanqi Chen, Zhaofei Yu, Wei Fang, Zhengyu Ma, Tiejun Huang, and Yonghong Tian. State transition of dendritic spines improves learning of sparse spiking neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 3701–3715, 2022.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.

Wei Fang, Zhaofei Yu, Yanqi Chen, Tiejun Huang, Timothée Masquelier, and Yonghong Tian. Deep residual learning in spiking neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 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), 2023. doi: 10.1126/sciadv.adi1480.

Wulfram Gerstner, Werner M. Kistler, Richard Naud, and Liam Paninski. *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. Cambridge University Press, 2014.

Ahmed Hasssan, Jian Meng, Anupreetham Anupreetham, and Jae-Sun Seo. SpQuant-SNN: Ultra-low precision membrane potential with sparse activations unlock the potential of on-device spiking neural networks applications. *Frontiers in Neuroscience*, 18:1440000, 2024. doi: 10.3389/fnins.2024.1440000.

Mark Horowitz. Computing's energy problem (and what we can do about it). In *IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 10–14, 2014. doi: 10.1109/ISSCC.2014.6757323.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. SNIP: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations (ICLR)*, 2019.

Wolfgang Maass. Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, 10(9):1659–1671, 1997. doi: 10.1016/S0893-6080(97)00011-7.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 54, pp. 1273–1282, 2017.

L.W. Meng, G.C. Qiao, X.Y. Zhang, J. Bai, Y. Zuo, P.J. Zhou, Y. Liu, and S.G. Hu. An efficient pruning and fine-tuning method for deep spiking neural network. *Applied Intelligence*, 53:28910–28923, 2023. doi: 10.1007/s10489-023-05056-8.

Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H. Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature Communications*, 9:2383, 2018. doi: 10.1038/s41467-018-04316-3.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. In *International Conference on Learning Representations (ICLR)*, 2017.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11264–11272, 2019.

Emre O. Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019. doi: 10.1109/MSP.2019.2931595.

Kerem Ozfatura, Emre Ozfatura, and Deniz Gündüz. Distributed sparse SGD with majority voting. *arXiv preprint arXiv:2011.06495*, 2020.

Jiangrong Shen, Qi Xu, Jian K. Liu, Yueming Wang, Gang Pan, and Huajin Tang. ESL-SNNs: An evolutionary structure learning strategy for spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 86–93, 2023. doi: 10.1609/aaai.v37i1.25079.

Nicolas Skatchkovsky, Hyeryung Jang, and Osvaldo Simeone. Federated neuromorphic learning of spiking neural networks for low-power edge intelligence. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8524–8528. IEEE, 2020. doi: 10.1109/ICASSP40776.2020.9053861.

Hidenori Tanaka, Daniel Kunin, Daniel L. K. Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

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. doi: 10.1609/aaai.v40i31.39787.

Chris Xing Tian, Yibing Liu, Haoliang Li, Ray C. C. Cheung, and Shiqi Wang. Gradient-congruity guided federated sparse training. In *ICLR 2024 Workshop on Privacy Regulation and Protection in Machine Learning*, 2024.

Yeshwanth Venkatesha, Youngeun Kim, Leandros Tassiulas, and Priyadarshini Panda. Federated learning with spiking neural networks. *IEEE Transactions on Signal Processing*, 69:6183–6194, 2021. doi: 10.1109/TSP.2021.3121632.

Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations (ICLR)*, 2020.

Junqiao Wang, Zhehang Ye, and Yuqi Ouyang. Spiking layer-adaptive magnitude-based pruning. *arXiv preprint arXiv:2603.14946*, 2026.

Yuan Wang, Shukai Duan, and Feng Chen. Efficient asynchronous federated neuromorphic learning of spiking neural networks. *Neurocomputing*, 557:126686, 2023. doi: 10.1016/j.neucom.2023.126686.

Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, Yuan Xie, and Luping Shi. Direct training for spiking neural networks: Faster, larger, better. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 1311–1318, 2019. doi: 10.1609/aaai.v33i01.33011311.

A Federated Training Procedure

At communication round t, the server broadcasts the current global weights and masks to all clients. Each client copies this model, trains the masked SNN on its private data for the prescribed local epochs, and records the membrane potentials and spike outputs needed for rewiring. If t belongs to the predefined rewiring schedule, the client computes the membrane gates and firing rates from these local traces, prunes active connections using membrane-gated importance, and grows the same number of inactive connections using activity-guided importance. Newly grown weights are initialized with the layer-specific scale, so every layer preserves its target density. The client then uploads its updated sparse weights and mask. After receiving all client updates, the server counts client support for each layer, forms the support-weighted raw average, and applies support-first server alignment to retain exactly K_l global positions before broadcasting the next global sparse model. Algorithm 1 summarizes this sequence. Membrane potentials and spike traces are used only for local rewiring and remain on the client; the server receives only model weights and masks.

B Experimental Setup Details

Two layers need a convention. The first convolution receives the static image rather than a spike train, so $|x|$ replaces presynaptic spikes when its growth score is accumulated. The linear readout has no LIF module of its own, so its gate is computed from an auxiliary LIF applied to the readout current, and its presynaptic trace is the pooled and flattened feature sequence that the readout actually consumes.

The initial learning rate was 0.1, decayed by a cosine schedule whose progress is measured in communication rounds rather than local epochs. Rewiring was applied at every round inside a fixed window, rounds 10 to 500 on CIFAR-10 and 10 to 800 on CIFAR-100, and the rewiring ratio π_t followed a cosine decay from 0.30 to 0.05 across that window. Newly grown weights were drawn from a Gaussian whose scale is the empirical standard deviation of the layer's currently active weights, floored at 10^{-3}, falling back to 0.02 when a layer has too few active weights for that estimate. Static images were presented to the first convolution at every one of the $T = 10$ simulation steps rather than rate-coded, and the membrane and spike traces used for rewiring were accumulated over the whole of that round's local training and averaged over samples before the topology update.

Baseline adaptation and ablation variants. ESL-SNN was designed for single-model training, so the federated comparison uses ESL-SNN-FL: every client retains its magnitude pruning and candidate-gradient growth and uploads its locally rewired weights and support, and the server forms a support-normalized weighted average over the clients that retain each coordinate. Each ablation variant replaces one rule and leaves the other two untouched. *w/o MGP* drops the membrane gate from Equation 3, leaving the plain Taylor score of Equation 1. *w/o AGG* replaces the co-activity score of Equation 8 with growth drawn uniformly at random from the inactive positions. *w/o SFA* replaces the support-first rule of Equation 12 with a Top-K_l selection on aggregated weight magnitude alone, which is the magnitude-led alternative that Section 3.3 argues against. *w/o (MGP+AGG)* applies both substitutions together while retaining support-first alignment. All ablation runs use $0.75\times$ density and are otherwise identical to the full method.

C Additional Analysis

C.1 Mechanistic Analysis of MADST

Why membrane-gated pruning helps. For a silent target neuron j receiving a source spike, the minimum positive weight change required to trigger firing is

$$\Delta w_{ij}^{\mathrm{fire}} = \tau_m \left(V_{\mathrm{th}} - u_j[t]\right). \tag{14}$$

A neuron closer to the threshold is therefore more sensitive to a connection change. The consistent degradation after removing MGP shows that threshold proximity carries ranking information beyond the loss sensitivity captured by the Taylor score.

Why activity-guided growth uses two neural states. Removing AGG produces the largest single drop in the ablation table, and the value of guided regrowth depends on both the observed spike activity and the candidate topology produced by pruning. The non-monotonic ordering of the partial variants places pruning and growth in a coupled rewiring process rather than as independent contributions.

Why support-first alignment matters. The larger degradation after removing SFA on CIFAR-100, particularly under non-IID data, shows that server-side support reconciliation matters more when client topologies are more diverse. The recorded mask dynamics show the same thing: in the CIFAR-100 IID run at $0.50\times$ density, the first rewiring update changes 24.51% of global mask positions, after which client-mask Jaccard similarity increases from 0.772 to 0.990 and the global mask-change rate decreases to 0.069% by round 800. These dynamics trace early client-specific exploration followed by convergence toward a stable shared support.

C.2 Communication

Each round a client uploads the $K_l = \lfloor \rho_l P_l \rfloor$ active values of every sparse layer together with the mask that locates them. Measured in model elements, of the 11.17×10^6 parameters of Spiking ResNet-18 on CIFAR-10, 11.16×10^6 lie in the 21 masked convolution and linear tensors and 9.6×10^3 in the normalization layers, which stay dense. The per-round payload is therefore ρ of the maskable parameters plus that fixed remainder: 2.80, 5.59 and 8.38×10^6 elements per client at $0.25\times$, $0.50\times$ and $0.75\times$ density, against 11.17×10^6 for dense federated training, reductions of 74.9%, 50.0% and 25.0%. Across the ten clients of the CIFAR-10 setting this is 55.86×10^6 elements per round at $0.50\times$ against 111.74×10^6 for dense, and the payload holds at that level for all 500 rounds. MADST adds nothing to this budget: membrane potentials and spike traces are used only to compute the local rewiring scores and are never uploaded, so the server receives exactly the sparse weights and mask. A design that instead reconciled client topologies by exchanging membrane statistics would pay for the membrane signal in bandwidth. Support-first alignment obtains it by voting over supports the clients already transmit.

C.3 Energy Analysis

SNNs are widely regarded as a low-power alternative to ANNs due to their event-driven, spike-based computation. To assess whether MADST's sparse topology preserves this energy advantage under federated non-IID training, we estimate per-inference energy by jointly accounting for weight density and empirically measured spike activity, and compare against an equivalent dense one-step ANN of the same architecture.

We instrument a representative client ($k = 0$) in the CIFAR-100 non-IID run and record the layer-wise firing rate r_{l-1} after the global model has converged. Let C_l denote the one-step dense MAC count of layer l, and ρ the uniform target weight density. Because our implementation presents the static input to the first convolution at each of the T simulation steps, the per-inference operation-level energy is

$$E_{\mathrm{SNN}} = T\rho C_1 E_{\mathrm{MAC}} + T\rho \sum_{l=2}^{L} r_{l-1} C_l E_{\mathrm{AC}}, \tag{15}$$

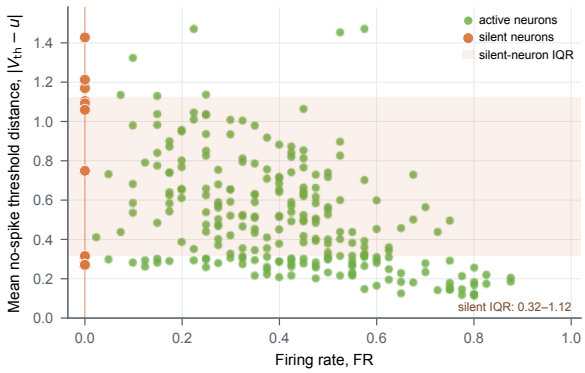


Figure 2: Threshold distance varies among neurons with the same firing rate. Each point is one postsynaptic neuron of a convolutional layer on CIFAR-100 non-IID data at $0.75\times$ density. The horizontal axis is the neuron's firing rate, and the vertical axis is its mean distance $V_{\text{th}} - u_j[t]$ over the time steps in which it did not spike.

whereas the equivalent one-step ANN requires

$$E_{\text{ANN}} = \sum_{l=1}^{L} C_l E_{\text{MAC}}. \tag{16}$$

Following prior operation-level energy estimates at 45 nm (Horowitz, 2014), we set $E_{\text{MAC}} = 4.6$ pJ and $E_{\text{AC}} = 0.9$ pJ.

To isolate the effect of weight density from spike activity, we hold r_{l-1} fixed at its converged value and vary only ρ across the three sparse budgets. The resulting per-inference energy is 0.049, 0.097 and 0.146 mJ at $0.25\times$, $0.50\times$ and $0.75\times$ density, respectively, against 2.555 mJ for the dense ANN baseline—reductions of 98.1%, 96.2% and 94.3%. Even at the least sparse of these budgets, where weight sparsity removes only a quarter of the parameters, event-driven computation accounts for most of this gap. At a measured firing workload, the operation-level cost of the sparse SNN is one to two orders of magnitude below the equivalent dense ANN.

D Firing Rate Does Not Determine Threshold Distance

E Preliminaries

Federated sparse SNN setting. We consider a federated learning system with N clients and one central server. Client k owns a private dataset $\mathcal{D}_k$. All clients use the same SNN architecture, and each trainable sparse layer l is represented by a weight tensor W_k^l and a binary mask

$$M_k^l \in \{0,1\}^{\text{shape}(W^l)}. \tag{17}$$

The effective weight used in forward and backward propagation is

$$\widetilde{W}_k^l = W_k^l \odot M_k^l, \tag{18}$$

where $\odot$ denotes element-wise multiplication. Each sparse layer has a prescribed density ρ_l. Let $P_l = |W^l|$ denote the total number of scalar parameters in layer l. The number of active connections is fixed as

$$K_l = \lfloor \rho_l P_l \rfloor, \qquad \|M_k^l\|_0 = K_l. \tag{19}$$

We call the active positions of a mask its sparse support:

$$\text{supp}(M_k^l) = \{e : M_k^l(e) = 1\}, \tag{20}$$

where e indexes a scalar connection, such as $e = (i,j)$ in a fully connected layer or $e = (o,i,p,q)$ in a convolutional layer.

At communication round t, the server broadcasts the current global sparse model (W^t, M^t) to the participating clients S_t. Each client trains the masked SNN on its local data. At scheduled topology-update points, client k removes q_l active connections from layer l and grows the same number of inactive connections, where

$$q_l = \lfloor \pi_t K_l \rfloor , \tag{21}$$

with π_t denoting the rewiring ratio. The prune-and-grow operation preserves the exact density:

$$\|M_k^{l,\text{before}}\|_0 = \|M_k^{l,\text{after}}\|_0 = K_l. \tag{22}$$

Due to heterogeneous data distributions and client-specific neuronal dynamics, local rewiring leads to inconsistent sparse supports across clients. This support divergence poses a challenge for federated sparse aggregation, requiring the server to reconcile heterogeneous masks while preserving the prescribed density of each layer.

Leaky Integrate-and-Fire (LIF) neuron. We use the discrete LIF model with a hard reset to zero. For a postsynaptic neuron j, the masked presynaptic input at time step t is

$$x_j[t] = \sum_i (w_{ij} m_{ij}) s_i[t], \tag{23}$$

where $m_{ij} \in \{0, 1\}$ is the mask value for connection (i, j) and $s_i[t]$ is the presynaptic spike. Let τ_m denote the membrane time constant and V_{th} the firing threshold. The pre-reset membrane potential, spike output, and post-reset membrane potential are:

$$u_j[t] = v_j[t-1] + \frac{1}{\tau_m} \left(x_j[t] - v_j[t-1] \right), \tag{24}$$

$$s_j[t] = \Theta(u_j[t] - V_{\text{th}}), \tag{25}$$

$$v_j[t] = u_j[t] \left(1 - s_j[t] \right). \tag{26}$$

Here Θ is the Heaviside step function, $u_j[t]$ is the pre-reset membrane potential used for the spike decision, and $v_j[t]$ is the membrane potential after reset. Let T denote the number of simulation steps. We use the following ATan surrogate derivative during backpropagation (Fang et al., 2021; 2023):

$$\sigma'(x) = \frac{\alpha}{2 \left(1 + \left(\frac{\pi \alpha x}{2} \right)^2 \right)}, \tag{27}$$

where α controls the surrogate slope. We treat τ_m as a time constant rather than as a multiplicative decay coefficient; the corresponding leak coefficient is $1 - 1/\tau_m$. The firing rate of neuron j is

$$\text{FR}(j) = \frac{1}{T} \sum_{t=1}^{T} s_j[t]. \tag{28}$$