Spike-SFT: Selective Parameter Enhancement and Fusion for Efficient Spiking Neural Networks

Xiubo Liang[1], **Jinxing Han**[1], **Hongyi Duan**[1], **Zhenxing Li**[1],
Yuke Li[2], **Haoqi Zhu**[2], **Yu Zhao**[1], **Hongzhi Wang**[1,2*]

[1] School of Software Technology, Zhejiang University, Ningbo, China
[2] NetEase Yidun AI Lab, Hangzhou, China

Abstract

Spiking Neural Networks (SNNs) promise energy-efficient inference, but adapting large pre-trained SNNs to new tasks is expensive because BPTT memory and runtime scale with the number of simulation steps T. We observe that weights in pre-trained SNNs are strongly concentrated near zero, and under thresholded dynamics many such synapses are potentially low-impact. Based on this, we propose **Spike-SFT**, a two-stage adaptation framework. Stage 1 (**Selective Parameter Enhancement**, SPE) fine-tunes only a small-magnitude subset via masked updates with low-rank regularization, progressive re-selection, and sparse-gradient storage, reducing memory while maintaining competitive accuracy. Stage 2 (**PickIt**) fuses multiple SPE-adapted models by alignment and interference-aware delta merging with spike-statistics calibration, yielding a single merged model without extra inference modules. Across several SNN backbones and benchmarks, Spike-SFT offers a favorable accuracy–cost trade-off, improving fine-tuning efficiency while retaining strong accuracy, and enabling single-checkpoint weight-space fusion via PickIt.

1 Introduction

Spiking Neural Networks (SNNs) represent information with temporally sparse spike trains and compute through event-driven synaptic dynamics, offering a promising path toward energy-efficient inference on neuromorphic hardware [1, 2, 3]. With increasingly large SNN backbones and the emergence of pre-trained checkpoints, fine-tuning SNNs for downstream tasks has become a practical necessity. However, SNN fine-tuning is expensive: backpropagation-through-time (BPTT) scales memory and runtime with the number of simulation steps T, and full fine-tuning additionally requires storing dense gradients and optimizer states for all parameters. Moreover, downstream adaptation is rarely a one-shot procedure—it is common to run multiple fine-tuning jobs and keep only the best checkpoint, discarding complementary knowledge learned by other runs.

This paper tackles two coupled questions: *(i)* how can we adapt large pre-trained SNNs to new tasks with substantially lower training memory and time, and *(ii)* how can we consolidate multiple fine-tuned SNNs into a single model without increasing inference cost? We propose **Spike-SFT**, a two-stage framework that addresses both. Our starting point is an empirical observation: examined layers of pre-trained spike-driven SNNs exhibit a strong concentration of weights near zero (Appendix Fig. 3). Under thresholded spiking dynamics, some such near-zero synapses may be weakly expressed, suggesting low-cost adaptation candidates whose influence depends on spike activity and membrane threshold margins (Sec. 3.1).

*Corresponding author: hongzhiwang@zju.edu.cn

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

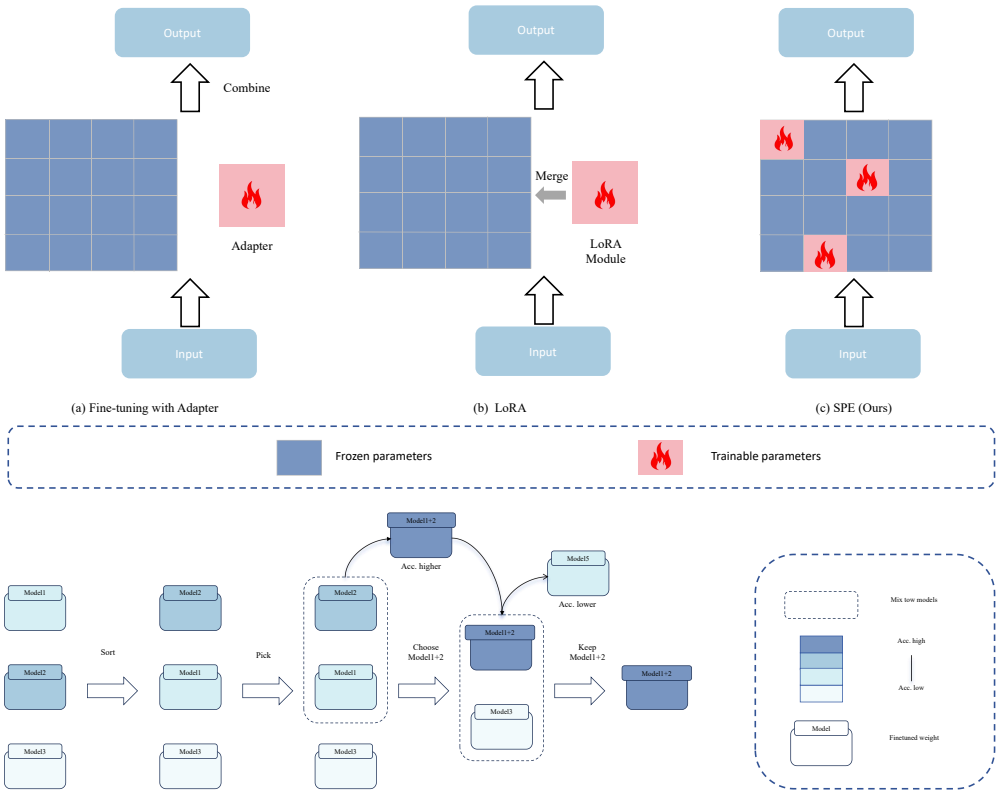

Figure 1: Overview of Spike-SFT. **Top:** comparison between Adapter, LoRA, and SPE in terms of frozen and trainable parameters. **Bottom:** PickIt fusion workflow for merging multiple SPE-adapted models.

Stage 1: Selective Parameter Enhancement (SPE). SPE performs parameter-efficient fine-tuning (PEFT) by updating only a masked subset of near-zero weights, together with progressive re-selection and low-rank regularization (Sec. 3.2). Crucially, SPE implements an unstructured sparse-gradient backpropagation that stores gradients and optimizer states only for the selected coordinates, directly reducing training memory under BPTT.

Stage 2: PickIt-based model fusion. Given multiple SPE-adapted models, PickIt fuses them in weight space into a single merged model without additional inference modules (Sec. 3.3). Within a domain, PickIt uses the same greedy selection loop depicted in Fig. 1 but replaces naive weight averaging with an interference-aware delta merger that filters conflicting and low-signal updates before aggregation. Across domains, PickIt additionally addresses permutation mismatch by parameter alignment and stabilizes spiking dynamics via spike-statistics calibration, mitigating activation-scale shifts after merging.

We evaluate Spike-SFT on competitive SNN backbones across both static-image and event-stream datasets, and further study cross-domain fusion on DomainNet (Sec. 4). Across settings, SPE matches or closely approaches full fine-tuning while substantially reducing training time and GPU memory (Tab. 1). PickIt further improves robustness in intra-domain fusion and yields balanced performance in cross-domain fusion (Tabs. 2, 3). In summary, the main contributions of this paper are:

- We identify a consistent near-zero weight concentration in large pre-trained spike-driven SNNs and motivate exploiting low-magnitude parameters for downstream adaptation.
- We propose SPE, a memory-efficient PEFT method for SNNs that updates only selected near-zero parameters, combines progressive re-selection with low-rank regularization, and uses sparse-gradient storage to reduce BPTT memory (Sec. 3.2).

- We develop PickIt, a training-free fusion procedure that merges multiple SPE-adapted models into a single model using interference-aware delta merging, cross-domain alignment, and spike-stable calibration, without additional inference modules (Sec. 3.3).
- We provide extensive evaluations across backbones and datasets, reporting both accuracy and practical training costs to quantify gains in large-scale SNN adaptation (Sec. 4).

2 Related Work

2.1 Spiking Neural Network Training

Training algorithms for SNNs are broadly categorized into unsupervised and supervised approaches. Unsupervised learning, inspired by biological plasticity, includes Hebbian learning [4] and Spike-Timing-Dependent Plasticity (STDP) [5]. Supervised learning is further divided into indirect and direct methods. Indirect methods such as ANN-to-SNN conversion [6, 7, 8] train an ANN first and then convert it into an SNN; while conceptually simple, conversion often requires careful tuning of SNN-specific hyperparameters and may incur accuracy loss. Direct supervised methods enable end-to-end optimization of SNNs using surrogate gradients, evolving from early methods like SpikeProp [9] to more general approaches such as STBP [10]. A key recent concern is that surrogate gradients may destroy the intrinsic sparsity of spike-based computation; recent work proposes maintaining sparsity during training by masking surrogate gradients and using temporally weighted decoding, improving generalization across architectures and datasets [11]. Recent work examines SNN thresholds, calibration, and membrane-state consistency [12, 13, 14]. In continuous-control reinforcement learning, Proxy Target shows that discrete spikes complicate ANN-designed target-network updates [15].

Alongside algorithmic advances, large-scale SNN architectures have rapidly evolved toward Transformer-style backbones. SpikingResformer bridges multi-stage ResNet-style inductive biases with spiking self-attention and reports strong ImageNet performance with few time steps [16]. More recently, SpiLiFormer introduces a lateral-inhibition inspired design to suppress irrelevant contexts in spiking attention and reports consistent gains across both frame-based and event-based benchmarks [17]. Beyond these visual benchmarks, Signal-SGN models skeleton sequences using spiking graph convolutions and temporal-frequency features [18]. In parallel, as SNN models scale, the community is beginning to explore pre-training then fine-tuning for SNNs: a recent NeurIPS workshop paper demonstrates self-supervised pre-training by contrasting paired events and frames, enabling transferable representations for downstream event-based classification [19]. These trends motivate our focus on efficient fine-tuning and stable fusion of multiple large pre-trained SNN checkpoints.

2.2 Parameter-Efficient Fine-Tuning

PEFT methods adapt large pre-trained models efficiently by updating only a small subset of parameters or by adding lightweight trainable modules. They are commonly grouped into: (i) Additive Fine-tuning (AFT), which inserts external modules such as Adapters [20, 21, 22, 23, 24]; (ii) Reparameterized Fine-tuning (RFT), which learns structured updates (e.g., LoRA) that can be merged back into weights for low-latency inference [25, 26, 27]; and (iii) Selective Fine-tuning (SFT), which updates a sparse subset of original parameters while freezing the rest [28, 29, 30]. A practical bottleneck of many SFT variants is that they still require dense gradient computation/optimizer states before sparsification. PaFi [29], SHiRA [31], and NanoAdam [32] examine magnitude-based selection, sparse adapter fusion, and dynamic small-weight tuning, respectively, in complementary model settings.

Very recent PEFT research has revisited selective updates from the optimization and representation perspective. SparseGrad selectively sparsifies MLP-layer gradients to reduce memory while remaining competitive with strong PEFT baselines [33]. For structured adapters, SeLoRA improves LoRA by spectral re-parameterization to reduce redundancy and boost performance under tighter parameter budgets [34], and Kron-LoRA explores Kronecker-structured factorization combined with low-rank compression for further parameter reductions [35]. Our Spike-SFT framework is closest in spirit to selective fine-tuning: it selects low-magnitude synaptic coordinates, but crucially addresses SNN-specific memory constraints via a dedicated backpropagation strategy tailored to spike dynamics.

2.3 Model Merging and Weight-Space Fusion

Model merging aims to combine multiple fine-tuned experts into a single model without additional gradient-based training, enabling multi-task or cross-domain generalization. Classic lines include ensembling/model soups [36], task-vector arithmetic [37], and training-free interference-aware fusion such as ZipIt [38]. Nowadays, the focus has shifted toward explicitly mitigating interference and localizing task information. EMR-Merging proposes a tuning-free pipeline that elects a unified model and attaches extremely lightweight task-specific masks/rescalers to better match each expert [39]. TALL-masks show that different tasks often rely on largely non-overlapping parameter supports; building on this, Consensus Merging removes "catastrophic" weights that harm fusion quality [40]. TIES-Merging trims small updates and elects coordinate-wise signs before merging sign-consistent values [41]. DELLA-Merging further reduces destructive interference via magnitude-based sampling and stochastic pruning during task-vector fusion [42]. Optimization-based views also emerged: APGD formulates multi-task merging as constrained optimization solved by adaptive projective gradient descent [43], while RegMean++ extends regression-based merging by incorporating cross-layer dependencies for better effectiveness and generalization [44].

3 Method

3.1 Motivation: Low-Magnitude Synapses in SNNs

Our empirical analysis of pre-trained spike-driven SNNs reveals a concentration pattern in the examined layers: the weight tensors exhibit a pronounced concentration around zero (see Fig. 3). To understand the implication of this distribution, we consider the dynamics of a LIF neuron, where each synapse contributes an incremental drive $\Delta u_i[t] = w_{ij}x_j[t]$ to the membrane potential. When the absolute weight $|w_{ij}|$ is negligible, this input contributes minimally to the membrane potential accumulation, not necessarily triggering a threshold crossing. Consequently, under the event-driven dynamics of SNNs, these abundant near-zero connections may be low-impact.

More concretely, when $x_j[t] \in \{0, 1\}$ denotes the pre-synaptic spike, the maximal cumulative drive contributed by a single synapse over T steps is bounded by $\sum_{t=1}^{T} |w_{ij}x_j[t]| \leq |w_{ij}| \sum_{t=1}^{T} x_j[t] \leq |w_{ij}|T$. With typical firing rate $r_j = \frac{1}{T}\sum_t x_j[t]$, the expected drive scales as $|w_{ij}|\, r_j T$. Thus, when $|w_{ij}|\, r_j T$ is far below the effective threshold margin, the connection may have limited impact, subject to leak and threshold proximity (Appendix A.2). This motivates selecting trainable parameters from the near-zero population and allocating adaptation capacity to update a selected fraction of these synapses.

We identify these weak connections as candidates for task adaptation. Modifying large-magnitude weights risks perturbing the established firing pathways, potentially destabilizing the pre-trained knowledge representations. In contrast, selectively strengthening specific near-zero synapses allows the network to adapt spike routing pathways for task-specific adaptation, while simultaneously leaving most pre-trained weights unchanged. This insight motivates our proposed framework, Spike-SFT, which operates in two stages: (i) updating a sparse subset of low-magnitude parameters (Stage 1, SPE), and (ii) fusing multiple adapted models into a unified weight set without additional inference modules (Stage 2, PickIt). We further evaluate this selection principle by comparing it against random and magnitude-based strategies in Sec. 4.

3.2 Selective Parameter Enhancement (SPE)

The first stage of Spike-SFT is SPE, a memory-efficient fine-tuning strategy tailored for the sparse dynamics of SNNs. SPE operates on the premise of adapting low-magnitude synapses—connections with near-zero weights that may have limited current impact but possess the potential to form new, task-specific spike pathways without disrupting the pre-trained knowledge representations.

3.2.1 Parameter Selection and Masking

Given a pre-trained weight tensor P_0, we identify potentially effective parameters by constructing a binary mask M based on a magnitude threshold θ_t:

$$M_{ij} = \begin{cases} 1 & \text{if } |(P_0)_{ij}| < \theta_t, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

In practice, we set θ_t per layer to control a fixed trainable ratio p:

$$\theta_t^{(l)} = \text{Quantile}\big(|P_0^{(l)}|,\, p\big), \qquad \frac{\|M^{(l)}\|_0}{|P_0^{(l)}|} \approx p, \tag{2}$$

which selects the bottom-p fraction of weights by magnitude within each layer and avoids scale-mismatch across layers.

In LIF-based SNNs, spikes are generated only when the membrane potential crosses a threshold. Synapses with negligible weights ($|w| \approx 0$) contribute minimally to this accumulation, which can limit their direct effect on membrane accumulation. Modifying these weights may reduce interference: it avoids perturbing the dominant pathways (large weights) that encode the foundational knowledge, while allowing the optimization process to update task-specific pathways.

3.2.2 Sparse Gradient Update with Low-Rank Constraint

During fine-tuning, updates are strictly confined to the masked subset via $\nabla P_{\text{trainable}} = \nabla P \odot M$. To encourage structured adaptation and suppress high-frequency/noisy updates that may destabilize spiking dynamics, we regularize the masked update $\Delta^{(l)} = P^{(l)} - P_0^{(l)}$ with a nuclear-norm penalty:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{task}} + \eta \sum_l \big\|\text{Mat}(\Delta^{(l)} \odot M^{(l)})\big\|_*. \tag{3}$$

Here $\|\cdot\|_*$ is the nuclear norm and $\text{Mat}(\cdot)$ reshapes the tensor to 2D. In implementation we approximate the nuclear norm with truncated (or randomized) SVD, and can compute it at a lower frequency (e.g., every few iterations) to keep overhead small.

3.2.3 Progressive Parameter Adjustment

A fixed mask may waste budget on coordinates that either (i) quickly grow and start behaving like strong pre-trained synapses, or (ii) remain near-zero and fail to absorb task information. PPA therefore performs a one-shot re-selection at an intermediate time $t = \gamma T_{\text{train}}$ (we use $\gamma = 0.5$ unless stated otherwise). Specifically, we (1) keep a subset of currently trainable weights that are still close to zero, and (2) refill the remaining budget with additional near-zero weights from the frozen pool to maintain the same trainable ratio p:

$$\mathcal{I} \leftarrow \textsc{Bottom}\big(|P|,\, p\big), \qquad v \leftarrow P[\mathcal{I}], \tag{4}$$

where BOTTOM selects bottom-p coordinates by magnitude (layer-wise). After re-selection, we re-pack the trainable vector v and restart (or remap) optimizer states on the new index set; cumulative support may exceed p. This "budget recycling" encourages SPE to continuously allocate capacity to low-impact coordinates and form new spike-routing pathways.

3.2.4 Memory-Efficient Unstructured Backpropagation

Naive masked fine-tuning still allocates dense optimizer states (and often dense gradient buffers) for all parameters, which is costly under BPTT. SPE re-parameterizes the trainable subset as a flat vector $v \in \mathbb{R}^{|\mathcal{I}|}$, where $\mathcal{I} = \{k \mid M_k = 1\}$ indexes masked coordinates. The effective weight is constructed on-the-fly by scattering v into frozen weights:

$$P = P_{\text{frozen}} + \text{Scatter}_{\mathcal{I}}(v). \tag{5}$$

Autograd then tracks gradients for v as the only leaf parameter, so persistent memory for gradients and optimizer states scales with $|\mathcal{I}|$ rather than $|P|$. For AdamW, this reduces parameter-side memory from storing $\{P, \nabla P, m, v\}$ at full size to storing $\{v, \nabla v, m_v, v_v\}$ for active coordinates (plus frozen P). Note that BPTT activation/state memory (scaling with simulation steps T) is orthogonal; SPE primarily reduces the parameter-side component; measured peak savings also reflect backward workspaces.

3.3 PickIt-based Model Fusion

The second stage of Spike-SFT leverages PickIt to fuse multiple SPE-adapted models into a single functional unit. Crucially, this fusion operates entirely in the weight space, aggregating knowledge from diverse models without adding inference-time branches or modules.

3.3.1 Intra-Domain Fusion

When fusing models fine-tuned on the same domain, directly averaging full weights is unnecessarily noisy because most coordinates do not carry consistent task-relevant updates. We therefore represent each SPE model as a task delta relative to the shared pre-trained initialization W_0: $W_i' = W_0 + \Delta_i$. We maintain a selected set $\mathcal{S}$ (initialized by the best validation model) and greedily test adding a candidate i by recomputing a merged delta:

$$\Delta_{\text{pool}} \leftarrow \mathcal{M}_{\text{CF}}\big(\{\Delta_j\}_{j\in\mathcal{S}\cup\{i\}}; \{a_j\}\big)\,, \tag{6}$$

$$W_{\text{pool}} = W_0 + \Delta_{\text{pool}}. \tag{7}$$

Recent training-free merging methods identify interference as the dominant failure mode, and mitigate it by (i) dropping low-signal updates (magnitude-based sampling / pruning) and (ii) eliminating "selfish/catastrophic" coordinates via mask agreement, before averaging. We implement a compact version that is compatible with SNN fusion and keeps architectural overhead at zero. For each layer, we keep only the top-τ fraction (by $|\Delta_j|$), excluding zero deltas, as a deterministic proxy of magnitude-based sampling, forming masks $M_j \in \{0,1\}^d$. Only positive-weight experts count toward κ. We then compute a support mask $M_{\text{cons}} = \mathbf{1}(\sum_j M_j \geq \kappa)$ (at least half, $\kappa = \lceil |\mathcal{S}|/2 \rceil$) and an elected sign $s = \text{sign}(\sum_j a_j\, M_j \odot \text{sign}(\Delta_j))$. The merged delta is

$$\begin{aligned} C_j &= M_j \odot \mathbf{1}[s \neq 0,\ \text{sign}(\Delta_j) = s]\,, \\ \mathcal{M}_{\text{CF}}(\{\Delta_j\}; \{a_j\}) &= M_{\text{cons}} \odot \frac{\sum_j a_j\, C_j \odot \Delta_j}{\sum_j a_j\, C_j + \epsilon}, \end{aligned} \tag{8}$$

where ϵ avoids division by zero; ties yield zero. In practice, we fix τ (e.g., 0.1) and use uniform a_j in same-domain pools. At endpoints, the sole expert is returned untrimmed.

3.3.2 Cross-Domain Alignment and Calibration

Fusing models from different domains is harder because permutation invariance and cross-domain conflicts amplify interference. We keep the same three-step structure (align $\rightarrow$ merge $\rightarrow$ calibrate), but replace naive linear interpolation by interference-aware delta merging.

1. Parameter alignment (activation-guided). We solve a linear assignment problem with the Jonker–Volgenant algorithm [45] to obtain a permutation Π and aligned weights $W_{B'} = \Pi W_B$. To improve robustness across domains, we compute the assignment cost on a small calibration set using channel-wise activation (or membrane-potential) correlations, rather than raw weight similarity.

2. Interference-aware delta merging. Let $W_A = W_0 + \Delta_A$ and define $\Delta_{B'} = W_{B'} - W_0$ after alignment; the latter need not remain sparse. We form the merged model by a conflict-filtered delta merge with weights $a_A = \alpha$ and $a_B = 1 - \alpha$:

$$W_{\text{merged}} = W_0 + \mathcal{M}_{\text{CF}}\big(\{\Delta_A, \Delta_{B'}\}; \{\alpha, 1-\alpha\}\big). \tag{9}$$

3. Spike-statistics calibration for membrane stability. Even after conflict filtering, merging may shift the activation scale. We therefore keep a data-light calibration step (no training): for each layer, estimate $\sigma_A = \text{std}(X_A)$, $\sigma_{B'} = \text{std}(X_{B'})$, and $\rho = \text{Corr}(X_A, X_{B'})$ on the calibration set, and rescale the merged pre-activations by

$$\beta = \frac{\alpha\sigma_A + (1-\alpha)\sigma_{B'}}{\sqrt{\alpha^2\sigma_A^2 + (1-\alpha)^2\sigma_{B'}^2 + 2\alpha(1-\alpha)\rho\,\sigma_A\sigma_{B'}}}. \tag{10}$$

This aims to restore membrane-potential dynamic range and mitigate diminished spiking. Eq. (10) follows by matching the variance of the linearly interpolated pre-activation $\text{Var}(\alpha X_A + (1-\alpha)X_{B'})$ (which expands with correlation ρ) to the desired convex combination $\alpha\sigma_A + (1-\alpha)\sigma_{B'}$; detailed derivation is provided in the appendix A.4.

Table 1: Performance and Efficiency Comparison on Static Image and Event Stream Datasets. Params (Trainable parameter percentages) for PEFT methods (e.g., SNN LoRA, SPE) vary across architectures due to differing total parameter counts and specific method configurations (e.g., LoRA rank).

Architecture	Method	Params	Static Image Datasets (%)		Event Stream Datasets (%)		Time	Memory
		(%)	CIFAR-10	CIFAR-100	DVS-CIFAR10	DVS128	(min)	(GB)
SEW-ResNet	FFT	100.00	95.23	76.58	74.12	97.29	121	10.54
	SNN LoRA	5.01	94.87	75.91	73.86	96.93	102	6.21
	SNN Adapter	6.08	94.71	75.67	73.74	96.81	107	6.83
	Random SFT	5.09	90.12	68.34	68.57	92.16	116	10.07
	Magnitude SFT	5.03	92.56	71.47	70.21	94.08	119	10.18
	SPE (Ours)	**5.06**	**95.19**	**76.43**	**74.08**	**97.17**	**93**	**5.29**
QKFormer	FFT	100.00	96.57	85.81	86.23	98.52	151	12.01
	SNN LoRA	6.03	96.18	85.17	86.04	98.41	132	7.05
	SNN Adapter	7.01	96.04	84.92	85.91	98.37	137	8.51
	Random SFT	6.02	91.81	78.03	80.16	94.89	146	11.52
	Magnitude SFT	6.04	93.58	80.27	82.31	96.19	149	11.64
	SPE (Ours)	**6.07**	**96.48**	**85.76**	**86.18**	**98.58**	**117**	**6.57**
Spike-driven	FFT	100.00	94.51	75.04	72.08	96.01	111	9.51
	SNN LoRA	4.02	94.13	74.37	71.52	95.54	91	5.53
	SNN Adapter	5.07	94.09	74.16	71.38	95.42	67	6.09
	Random SFT	4.01	89.06	67.09	66.53	91.05	106	9.02
	Magnitude SFT	4.09	91.52	70.01	69.04	93.07	109	9.24
	SPE (Ours)	**4.03**	**94.49**	**74.81**	**71.97**	**95.98**	**53**	**4.56**
Spike-driven v2	FFT	100.00	95.02	76.03	73.51	96.84	117	10.06
	SNN LoRA	4.54	94.68	75.37	72.95	96.31	97	6.02
	SNN Adapter	5.56	94.54	75.12	72.78	96.27	73	6.54
	Random SFT	4.58	89.57	68.01	67.89	91.52	113	9.57
	Magnitude SFT	4.51	92.09	71.03	69.87	93.54	115	9.71
	SPE (Ours)	**4.59**	**94.91**	**75.89**	**73.46**	**96.78**	**58**	**5.03**
Spikformer	FFT	100.00	96.01	84.06	85.58	98.09	141	11.02
	SNN LoRA	5.57	95.63	83.38	85.02	97.51	123	6.55
	SNN Adapter	6.54	95.59	83.19	84.83	97.46	127	8.03
	Random SFT	5.58	90.52	76.07	78.01	93.04	136	10.59
	Magnitude SFT	5.52	92.54	79.03	81.06	95.07	139	10.71
	SPE (Ours)	**5.51**	**95.98**	**83.81**	**85.49**	**97.92**	**68**	**5.21**
Spikformer v2	FFT	100.00	96.89	86.21	86.87	98.83	161	12.56
	SNN LoRA	6.52	96.41	85.59	86.32	98.35	143	7.51
	SNN Adapter	7.53	96.38	85.31	86.14	98.21	147	9.08
	Random SFT	6.51	91.57	78.52	80.53	94.57	156	12.04
	Magnitude SFT	6.58	93.81	81.56	83.09	96.52	159	12.27
	SPE (Ours)	**6.59**	**96.71**	**86.09**	**86.78**	**98.71**	**78**	**5.83**

4 Experiments

4.1 SPE: Accuracy and Training Efficiency

Tab. 1 reports accuracy and training cost across static-image and event-stream datasets. Across backbones, **SPE** achieves accuracy comparable to full fine-tuning while training only about 4–7% of parameters at a time. Compared to FFT, SPE consistently reduces peak GPU memory and often shortens training time, consistent with storing gradients and optimizer states only for the sparse trainable subset, alongside reduced backward workspaces (Appendix C). Under the same parameter budget, random and largest-magnitude SFT baselines are substantially worse, indicating that our small-magnitude selection criterion is useful for adaptation.

4.2 PickIt: Model Fusion Evaluation

Fig. 2(a) shows the results of three models with different architectures pre-trained on ImageNet and fine-tuned on CIFAR-100. As illustrated in the figure, accuracy increases with the number of models. However, once the number of models exceeds 5, the accuracy improvement slows down and tends to saturate. This is consistent with the greedy acceptance rule in Eq. (6): as the pool grows, newly added models become increasingly redundant, so gains naturally saturate. Fig. 2(b) illustrates the performance of the PickIt algorithm in comparison to individual models with various hyperparameters

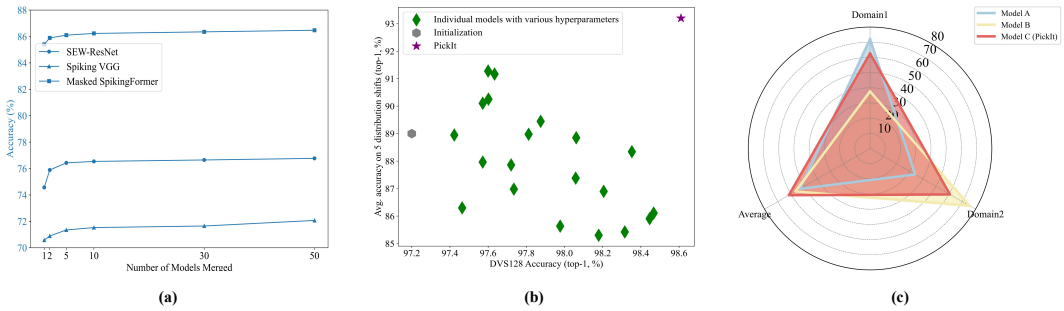

Figure 2: Experimental results visualizations. (a) Relationship between the number of models merged and accuracy for different architectures (SEW-ResNet, Spiking VGG, and Masked SpikingFormer). Accuracy improves as more models are merged, with gains tapering off after five models. (b) PickIt algorithm's performance in fine-tuning Meta-SpikeFormer on DVS128, where the purple star represents the highest accuracy achieved by PickIt, demonstrating superior generalization across distribution shifts. (c) Performance comparison of Model A, Model B, and Model C (PickIt) across two domains (Domain 1: Clipart and Domain 2: Painting), showing how Model C effectively balances accuracy across both domains.

and initialization models when fine-tuning Meta-SpikeFormer on the DVS 128 dataset. The purple star, representing the PickIt model, achieves the highest DVS128 accuracy at 98.61% and the highest average accuracy on distribution shifts at 93.2%. The spread of green diamonds indicates variability in the accuracy of individual models due to different hyperparameters. While some individual models perform well, others do not, showing inconsistency. The average accuracy on distribution shifts is a critical measure of how well a model generalizes to different data distributions. The PickIt model's superior performance in this aspect indicates its enhanced generalization capability, making it more robust against variations in data.

Fig. 2(c) shows the results of SEW-ResNet, pre-trained on the ImageNet dataset and then fine-tuned on the Clipart (Domain1) and Painting (Domain2) domains, with additional domain results available in the supplementary material. As illustrated, Model A (fine-tuned on Domain 1) excels in Domain 1 but underperforms in Domain 2, whereas Model B (fine-tuned on Domain 2) shows the opposite trend. Model C, created by merging Models A and B using the PickIt algorithm, balances performance across both domains and achieves higher average accuracy, demonstrating improved generalization across different data distributions.

4.2.1 Intra-Domain Fusion

We evaluate whether fusing multiple SPE-tuned models for the same task yields consistent gains over selecting the best run. As shown in Tab. 2, PickIt improves CIFAR-10 accuracy from 95.14 (best individual SPE) to 95.35, suggesting that greedy conflict-filtered delta pooling can consolidate complementary updates within a domain.

Table 2: Intra-domain fusion performance on CIFAR-10 using SPE fine-tuned models.

Method	CIFAR-10
Best Individual SPE	95.14
Average of Top 5 SPE	95.17
PickIt-Fused SPE (Ours)	**95.35**

4.2.2 Cross-Domain Fusion

As shown in Tab. 3, we evaluate cross-domain fusion on DomainNet by first constructing multiple SPE experts fine-tuned on different source domains (Clipart, Painting, Real, and Sketch). These single-domain experts exhibit strong specialization: each expert performs best on its own training domain but shows pronounced degradation on other domains, indicating limited cross-domain generalization. We then apply PickIt's cross-domain fusion to merge pairs of domains (Clipart+Painting, Real+Sketch) and also all four domains into a single model. Across settings, the fused model achieves a substantially more balanced accuracy profile over the fused domains, improving the average accuracy relative to naive reliance on a single specialized expert.

Table 3: Cross-Domain Fusion Performance on DomainNet Subsets.

Method	Clipart	Painting	Real	Sketch	Average
SPE fine-tuned on Clipart only	**85.21**	45.13	58.35	49.66	59.59
SPE fine-tuned on Painting only	48.92	**78.58**	55.84	52.15	58.87
SPE fine-tuned on Real only	59.14	52.37	**81.72**	56.40	62.41
SPE fine-tuned on Sketch only	51.79	55.00	59.89	**79.23**	61.48
PickIt-Fused SPE (Clipart + Painting)	80.50	76.18	-	-	**78.34**
PickIt-Fused SPE (Real + Sketch)	-	-	80.10	77.55	**78.83**
PickIt-Fused SPE (All 4 Domains)	78.90	75.80	79.55	76.10	**77.59**

Table 4: SPE component and selection ablations on SEW-ResNet/CIFAR-10.

Method / Setting	Parameter Selection Rule	Acc. (%)
SPE (Ours, Full)	Smallest Magnitude (θ_t)	**95.19**
SPE *w/o* L_{rank}	Smallest Magnitude (θ_t)	94.87
SPE *w/o* PPA	Smallest Magnitude (θ_t)	94.91
SPE (Random)	Random	90.13
SPE (Largest Magnitude)	Largest Magnitude	92.56
Traditional SFT Baseline	Mask Size Matched to SPE	95.08

These gains are consistent with the design of our fusion pipeline. First, parameter alignment is essential to establish a shared functional layout before merging; without alignment, the fused model suffers destructive interference due to permutation mismatch. Second, conflict-filtered delta merging aggregates task-relevant updates while suppressing conflicting coordinates, improving stability compared to naive interpolation. Finally, spike-stable calibration mitigates distributional shifts in activation scale that would otherwise collapse membrane potential variance and reduce firing activity. Importantly, the result of PickIt is a single merged checkpoint for deployment, so the improved cross-domain generalization comes with no architectural inference overhead.

4.3 Ablation Studies

4.3.1 SPE Components and Selection

To quantify the contribution of each component in SPE, we performed ablation studies on the nuclear-norm regularizer (L_{rank}) and the PPA schedule. We further benchmarked our low-magnitude parameter selection strategy against random and largest-magnitude selection heuristics under an identical parameter budget. Tab. 4 summarises the results. Removing L_{rank} or the PPA schedule degrades accuracy by 0.32 and 0.28 percentage points, respectively, supporting their contributions to adaptation accuracy. More critically, substituting our small-magnitude selection with random or largest-magnitude rules leads to drops of 2.63–5.06 percentage points. This suggests that small-magnitude weights are useful candidates for adaptation. While SPE approaches full fine-tuning accuracy, it drastically reduces the GPU memory footprint. This is attributed to our unstructured backpropagation implementation (Sec. 3.2), which stores gradients exclusively for the sparse trainable subset rather than the full tensor; backward-workspace differences also contribute (Appendix C.1).

4.3.2 PickIt Alignment and Calibration

As shown in Tab. 5, ablating the parameter-alignment stage—which enforces a shared functional layout before weight-space merging—leads to dramatic accuracy decreases of 15.32 and 7.62 percentage points, respectively. This suggests that naïvely fusing unaligned subnetworks introduces destructive interference. In contrast, disabling the subsequent variance-correction step results in smaller (1.67–2.39 point) losses, indicating that statistical matching refines but does not solely enable cross-domain transfer. Sensitivity to the mixing coefficient α is reported in Appendix Tab. 7.

Table 5: Ablation of PickIt Cross-Domain Components on DomainNet Fusion (Clipart & Painting).

Fusion Method	Clipart Acc. (%)	Painting Acc. (%)
PickIt-Fused SPE (Full)	**80.53**	**76.19**
w/o Parameter Alignment	65.21	68.57
w/o Variance Correction	78.14	74.52

5 Conclusion

We presented Spike-SFT, a two-stage framework for efficient adaptation and checkpoint fusion in pre-trained SNNs. SPE exploits the concentration of near-zero weights in spike-driven backbones and re-parameterizes the selected coordinates as sparse trainable vectors, reducing parameter-side gradient and optimizer-state memory while preserving accuracy close to full fine-tuning. PickIt then merges multiple SPE-adapted checkpoints in weight space through conflict-filtered delta aggregation, with alignment and spike-statistics calibration for cross-domain fusion. Experiments across static-image, event-stream, and DomainNet settings show that Spike-SFT provides a favorable accuracy–cost trade-off and enables single-checkpoint fusion without additional inference modules. Future work will extend the framework to broader SNN architectures and validate deployment-level energy savings on neuromorphic hardware.

6 Acknowledgement

This work was partly supported by Ningbo Key R&D Program (2025Z047) and Ningbo Major Application Demonstration Program (2025Z199).

References

[1] Wei Fang, Zhaofei Yu, Yanqi Chen, Tiejun Huang, Timothée Masquelier, and Yonghong Tian. Deep residual learning in spiking neural networks. *Advances in Neural Information Processing Systems*, 34:21056–21069, 2021.

[2] Zhaokun Zhou, Yuesheng Zhu, Chao He, Yaowei Wang, Shuicheng Yan, Yonghong Tian, and Li Yuan. Spikformer: When spiking neural network meets transformer. *arXiv preprint arXiv:2209.15425*, 2022.

[3] Man Yao, Jiakui Hu, Zhaokun Zhou, Li Yuan, Yonghong Tian, Bo Xu, and Guoqi Li. Spike-driven transformer. *Advances in neural information processing systems*, 36:64043–64058, 2023.

[4] Donald Olding Hebb. *The organization of behavior: A neuropsychological theory*. Psychology press, 2005.

[5] Natalia Caporale and Yang Dan. Spike timing–dependent plasticity: a hebbian learning rule. *Annu. Rev. Neurosci.*, 31:25–46, 2008.

[6] Abhronil Sengupta, Yuting Ye, Robert Wang, and Kaushik Roy. Going deeper in spiking neural networks: Vgg and residual architectures. *Frontiers in neuroscience*, 13:425055, 2019.

[7] Yongqiang Cao, Yang Chen, and Deepak Khosla. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 113:54–66, 2015.

[8] Qiang Yu, Chenxiang Ma, Shiming Song, Gaoyan Zhang, Jianwu Dang, and Kay Chen Tan. Constructing accurate and efficient deep spiking neural networks with double-threshold and augmented schemes. *IEEE Transactions on Neural Networks and Learning Systems*, 33(4):1714–1726, 2021.

[9] Sander M Bohte, Joost N Kok, and Johannes A La Poutré. Spikeprop: backpropagation for networks of spiking neurons. In *ESANN*, volume 48, pages 419–424. Bruges, 2000.

[10] 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.

[11] Yang Li, Feifei Zhao, Dongcheng Zhao, and Yi Zeng. Directly training temporal spiking neural network with sparse surrogate gradient. *Neural Networks*, 179:106499, 2024.

[12] Wei Liu, Li Yang, Mingxuan Zhao, Shuxun Wang, Jin Gao, Wenjuan Li, Bing Li, and Weiming Hu. Deeptage: Deep temporal-aligned gradient enhancement for optimizing spiking neural networks. In *International Conference on Learning Representations*, 2025.

[13] Jiaqi Yan, Changping Wang, De Ma, Huajin Tang, Qian Zheng, and Gang Pan. Training high performance spiking neural network by temporal model calibration. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pages 70289–70308. PMLR, 2025.

[14] Yongqi Ding, Lin Zuo, Mengmeng Jing, Pei He, and Hanpu Deng. Rethinking spiking neural networks from an ensemble learning perspective. In *International Conference on Learning Representations*, 2025.

[15] Zijie Xu, Tong Bu, Zecheng Hao, Jianhao Ding, and Zhaofei Yu. Proxy target: Bridging the gap between discrete spiking neural networks and continuous control. *Advances in Neural Information Processing Systems*, 38:159158–159184, 2026.

[16] Xinyu Shi, Zecheng Hao, and Zhaofei Yu. Spikingresformer: Bridging resnet and vision transformer in spiking neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5610–5619, 2024.

[17] Zeqi Zheng, Yanchen Huang, Yingchao Yu, Zizheng Zhu, Junfeng Tang, Zhaofei Yu, and Yaochu Jin. Spiliformer: Enhancing spiking transformers with lateral inhibition. *arXiv preprint arXiv:2503.15986*, 2025.

[18] Naichuan Zheng, Yuchen Du, Hailun Xia, and Zeyu Liang. Signal-sgn: A spiking graph convolutional network for skeleton action recognition via learning temporal-frequency dynamics. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 1539–1548, 2025.

[19] Raghav Singhal, Jan Finkbeiner, and Emre Neftci. Self-supervised pre-training of spiking neural networks by contrasting events and frames. In *UniReps: 2nd Edition of the Workshop on Unifying Representations in Neural Models*, 2024.

[20] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR, 2019.

[21] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. Adapterfusion: Non-destructive task composition for transfer learning. *arXiv preprint arXiv:2005.00247*, 2020.

[22] Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. *Advances in Neural Information Processing Systems*, 35:16664–16678, 2022.

[23] Tao Lei, Junwen Bai, Siddhartha Brahma, Joshua Ainslie, Kenton Lee, Yanqi Zhou, Nan Du, Vincent Zhao, Yuexin Wu, Bo Li, et al. Conditional adapters: Parameter-efficient transfer learning with fast inference. *Advances in Neural Information Processing Systems*, 36:8152–8172, 2023.

[24] Ali Edalati, Marzieh Tahaei, Ivan Kobyzev, Vahid Partovi Nia, James J Clark, and Mehdi Rezagholizadeh. Krona: Parameter efficient tuning with kronecker adapter. *arXiv preprint arXiv:2212.10650*, 2022.

[25] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[26] Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. Dylora: Parameter efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. *arXiv preprint arXiv:2210.07558*, 2022.

[27] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *International Conference on Learning Representations*. Openreview, 2023.

[28] Demi Guo, Alexander M Rush, and Yoon Kim. Parameter-efficient transfer learning with diff pruning. *arXiv preprint arXiv:2012.07463*, 2020.

[29] Baohao Liao, Yan Meng, and Christof Monz. Parameter-efficient fine-tuning without introducing new latency. *arXiv preprint arXiv:2305.16742*, 2023.

[30] Alan Ansell, Edoardo Maria Ponti, Anna Korhonen, and Ivan Vulić. Composable sparse fine-tuning for cross-lingual transfer. *arXiv preprint arXiv:2110.07560*, 2021.

[31] Kartikeya Bhardwaj, Nilesh Prasad Pandey, Sweta Priyadarshi, Viswanath Ganapathy, Rafael Esteves, Shreya Kadambi, Shubhankar Borse, Paul Whatmough, Risheek Garrepalli, Mart Van Baalen, et al. Sparse high rank adapters. In *NIPS*, 2024.

[32] Chao Zhou, Tom Jacobs, Advait Gadhikar, and Rebekka Burkholz. Pay attention to small weights. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

[33] Viktoriia A Chekalina, Anna Rudenko, Gleb Mezentsev, Aleksandr Mikhalev, Alexander Panchenko, and Ivan Oseledets. Sparsegrad: A selective method for efficient fine-tuning of mlp layers. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 14929–14939, 2024.

[34] Jiashun Cheng, Aochuan Chen, Nuo Chen, Ziqi Gao, Yuhan Li, Jia Li, and Fugee Tsung. Revisiting lora through the lens of parameter redundancy: Spectral encoding helps. *arXiv preprint arXiv:2506.16787*, 2025.

[35] Yixin Shen. Kronecker-lora: hybrid kronecker-lora adapters for scalable, sustainable fine-tuning. *arXiv e-prints*, pages arXiv–2508, 2025.

[36] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pages 23965–23998. PMLR, 2022.

[37] Samuel K Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries. *arXiv preprint arXiv:2209.04836*, 2022.

[38] George Stoica, Daniel Bolya, Jakob Brandt Bjorner, Pratik Ramesh, Taylor Hearn, and Judy Hoffman. Zipit! merging models from different tasks without training. In *International Conference on Learning Representations*, 2024.

[39] Chenyu Huang, Peng Ye, Tao Chen, Tong He, Xiangyu Yue, and Wanli Ouyang. Emr-merging: Tuning-free high-performance model merging. *Advances in Neural Information Processing Systems*, 37:122741–122769, 2024.

[40] Ke Wang, Nikolaos Dimitriadis, Guillermo Ortiz-Jimenez, François Fleuret, and Pascal Frossard. Localizing task information for improved model merging and compression. *arXiv preprint arXiv:2405.07813*, 2024.

[41] Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[42] Pala Tej Deep, Rishabh Bhardwaj, and Soujanya Poria. Della-merging: Reducing interference in model merging through magnitude-based sampling. *arXiv preprint arXiv:2406.11617*, 2024.

[43] Yongxian Wei, Anke Tang, Li Shen, Zixuan Hu, Chun Yuan, and Xiaochun Cao. Modeling multi-task model merging as adaptive projective gradient descent. *arXiv preprint arXiv:2501.01230*, 2025.

[44] The-Hai Nguyen, Dang Huu-Tien, Takeshi Suzuki, and Le-Minh Nguyen. Regmean++: Enhancing effectiveness and generalization of regression mean for model merging. *arXiv preprint arXiv:2508.03121*, 2025.

[45] Francesc Serratosa. Computation of graph edit distance: Reasoning about optimality and speed-up. *Image and Vision Computing*, 40:38–48, 2015.

[46] Peter Dayan and Laurence F Abbott. *Theoretical neuroscience: computational and mathematical modeling of neural systems*. MIT press, 2005.

[47] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[48] Hongmin Li, Hanchao Liu, Xiangyang Ji, Guoqi Li, and Luping Shi. Cifar10-dvs: an event-stream dataset for object classification. *Frontiers in neuroscience*, 11:309, 2017.

[49] Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo Di Nolfo, Tapan Nayak, Alexander Andreopoulos, Guillaume Garreau, Marcela Mendoza, et al. A low power, fully event-based gesture recognition system. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7243–7252, 2017.

[50] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1406–1415, 2019.

A Preliminary

A.1 Spike Neuron

In SNNs, the fundamental computational units are spike neurons. These neurons emit spikes contingent upon reaching a specific threshold, which is a key feature that distinguishes them from traditional artificial neurons. In our study, we utilize Leaky Integrate-and-Fire (LIF) neurons [46], which operate under a biologically inspired mechanism. The dynamics of an LIF neuron can be described by the following equations:

$$U[t] = V[t-1] + \frac{1}{k_\tau}\left(X[t] - (V[t-1] - V_{reset})\right), \tag{11}$$

$$S[t] = \mathcal{H}(U[t] - V_{th}), \tag{12}$$

$$V[t] = U[t](1 - S[t]) + V_{reset}S[t], \tag{13}$$

here, $U[t]$ represents the membrane potential at time t, $V[t-1]$ is the membrane potential at the previous time step, and $X[t]$ is the input at time t. The parameters k_τ, V_{th}, and V_{reset} signify the decay factor, firing threshold, and reset membrane potential, respectively. The Heaviside step function $\mathcal{H}(\cdot)$ is used to determine whether the membrane potential has crossed the firing threshold, resulting in a spike output $S[t]$.

A.2 Conditional effect of low-magnitude synapses

The bound in Sec. 3.1 concerns synaptic drive, not a universal guarantee of functional silence. In the LIF model of Eq. (11), let $\lambda = 1 - 1/k_\tau \in [0, 1)$ and remove a single synapse w_{ij}, holding other inputs and presynaptic spikes fixed. Until a spike/reset decision differs between the reference and perturbed trajectories, the resulting pre-reset membrane-potential perturbation obeys

$$|\delta U_i[t]| \le \frac{|w_{ij}|}{k_\tau}\sum_{k=1}^{t}\lambda^{t-k}x_j[k] \le \frac{|w_{ij}|(1-\lambda^t)}{k_\tau(1-\lambda)}. \tag{14}$$

A shared reset can only reduce this bound. If the right-hand side is smaller than the reference threshold margin $|U_i^{(0)}[t] - V_{\text{th}}|$ at every step, identical spike decisions follow by induction, under these fixed-input assumptions. This is a sufficient, not necessary, condition. The perturbations of multiple modified synapses can accumulate, so weight magnitude alone does not establish synaptic silence. SPE uses it as a low-cost selection proxy without requiring membrane traces or gradient saliency.

A.3 Permutation invariance

In typical neural architectures, neurons within each layer can be rearranged without altering the network's functionality; this characteristic is known as permutation invariance. In a basic feedforward network, this implies that the L-th weight matrix can be replaced by PW_L, and the $(L+1)_{th}$ weight matrix by $W_{L+1}P^{-1}$, without affecting the network's output. Consequently, even if two networks, θ_1 and θ_2, have learned the same functional set of neurons at each layer, these neurons might still be permuted or misaligned in different ways across the layers.

A.4 Variance Problem Definition

The variance problem occurs when two networks, even if functionally identical, exhibit a correlation between aligned units that is less than 1. Typically, for aligned networks, the correlation $\text{corr}(X_1, X_2)$ often significantly below 1 (e.g., around 0.4 in our setting). When considering a midpoint interpolated network (i.e., $\alpha = 0.5$), the variance of a unit X_α in the interpolated network is given by:

$$\text{Var}(X_\alpha) = \frac{\text{Var}(X_1) + \text{Var}(X_2) + 2\text{Cov}(X_1, X_2)}{4}. \tag{15}$$

This can be further expressed as:

$$\text{Var}(X_\alpha) = \left(\frac{\text{std}(X_1) + \text{std}(X_2)}{2}\right)^2 - \frac{(1 - \text{corr}(X_1, X_2))}{2} \cdot \text{std}(X_1) \cdot \text{std}(X_2)$$

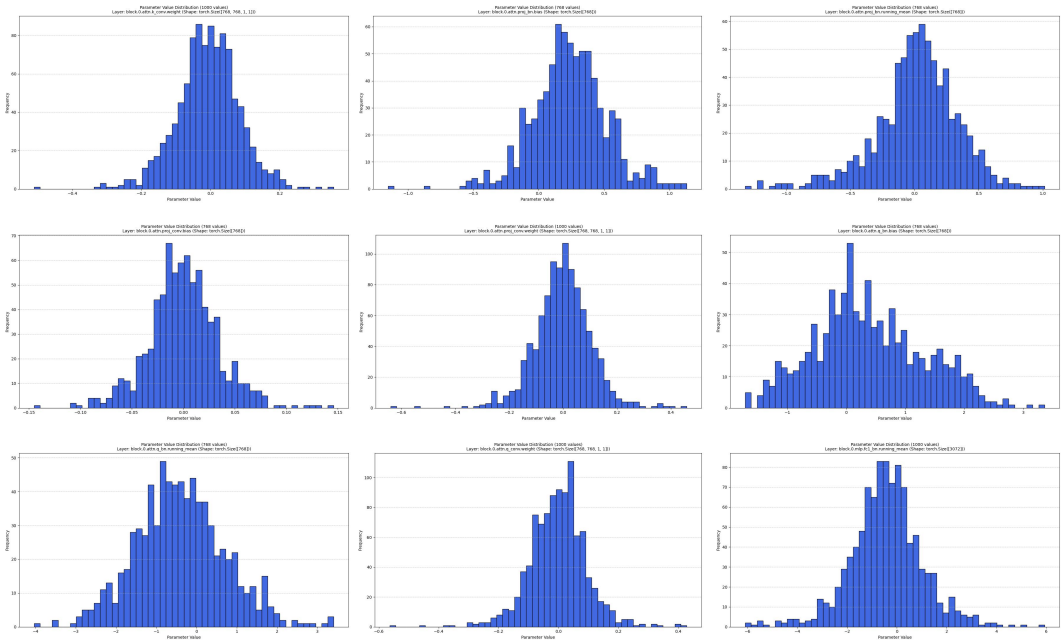

Figure 3: Weight distributions of Spike-Driven Transformer. Histograms across various layers illustrate that the parameter values are approximately distributed following a Gaussian function, with a significant concentration around zero.

Given that $\text{std}(X_1) \approx \text{std}(X_2)$, this simplifies to:

$$\text{Var}(X_\alpha) = (0.5 + 0.5 \cdot \text{corr}(X_1, X_2)) \cdot \text{Var}(X_1), \tag{16}$$

with $\text{corr}(X_1, X_2) \approx 0.4$, this results in:

$$\text{Var}(X_\alpha) = 0.7 \cdot \text{Var}(X_1), \tag{17}$$

this represents a 30% reduction in variance compared to the original network. This variance reduction, or 'variance collapse,' can become more pronounced in deeper layers of the network, leading to decreased model performance.

B Additional Experiments

Table 6: SPE Sensitivity to Threshold θ_t on CIFAR-10 Fine-tuning (SEW-ResNet, λ_{rank} fixed).

Threshold θ_t	Trainable Params (%)	CIFAR-10 Acc. (%)	GPU Memory (GB)
Small (10^{-4})	1.00	94.50	**5.0**
Medium (Our Setting)	5.00	**95.15**	5.8
Large (10^{-2})	15.00	95.05	7.1

B.1 Datasets

We utilize a diverse range of benchmark datasets to assess Spike-SFT under various conditions. For standard image classification and intra-domain fine-tuning evaluation, we use CIFAR-10 [47], CIFAR-100 [47], and ImageNet subsets. To evaluate performance on event-stream data and Spike-driven SNNs, we employ DVS-CIFAR10 [48] and DVS128 Gesture [49]. For cross-domain fusion experiments, we leverage the DomainNet dataset [50], comprising images from distinct domains (e.g., Clipart, Painting, Real, Sketch), which allows us to train models on different source domains and evaluate their fusion.

B.2 Architectures

To demonstrate the broad applicability of Spike-SFT, we conduct experiments on several state-of-the-art spike-driven SNN architectures. This includes variants of Spiking Residual Networks, such as SEW-ResNet [1], and Spiking Transformers [2, 3], which have shown competitive performance on complex vision tasks. We utilize pre-trained checkpoints of these architectures obtained from publicly available sources or trained following established protocols.

B.3 Training Details

Unless stated otherwise, we keep the same simulation steps T as the pre-trained checkpoint and train with AdamW (cosine LR schedule). We report the best validation checkpoint for each run. For SPE, we set the trainable ratio p (reported in Tab. 1), use layer-wise bottom-p magnitude selection (Eq. 2), apply PPA once at $t = \gamma T_{\text{train}}$ (default $\gamma = 0.5$), and use a fixed rank-regularization weight η. For PickIt, we use task deltas relative to the same W_0, apply conflict filtering with a fixed top-τ ratio and majority threshold κ, and perform cross-domain alignment/calibration on a small calibration split $\mathcal{D}_{cal}$ (a small subset of training data).

B.4 Baselines

We compare Spike-SFT against several relevant baselines to highlight its advantages:

- **Full Fine-Tuning (FFT):** Fine-tuning all parameters of the pre-trained SNN. This represents the performance upper bound but is computationally expensive.
- **SNN LoRA:** We adapt the standard LoRA method [25] for SNNs, applying low-rank updates to weight matrices in selected layers. We test with varying ranks to find a competitive configuration.
- **SNN Adapter:** We adapt Adapter-based tuning [20] by inserting trainable adapter modules into the SNN architecture.
- **Traditional Selective Fine-Tuning (SFT):** We implement baseline SFT methods for SNNs, including:
  - *Random SFT:* Randomly selecting a fixed percentage of parameters to fine-tune.
  - *Magnitude SFT (Largest):* Selecting parameters with the largest absolute values for fine-tuning.

  These SFT baselines are implemented using a standard masking approach that requires computing gradients for all parameters, serving as a comparison for SPE's unstructured backpropagation memory efficiency.

We also evaluate the performance of the individual SPE fine-tuned models (Stage 1 in isolation) before the fusion stage to specifically assess SPE's efficiency and effectiveness as a standalone fine-tuning method compared to other baselines.

B.5 Evaluation Metrics

We employ the following metrics to evaluate model performance and efficiency:

- **Accuracy:** Top-1 or Top-5 classification accuracy on the respective test sets.
- **Average Accuracy (for Cross-Domain):** Mean accuracy across multiple source or target domains.
- **Trainable Parameters:** The number of parameters updated during fine-tuning, reported as a percentage of the total model parameters.
- **Training Time:** Run-level wall-clock time in minutes (not per-iteration latency).
- **GPU Memory Usage:** Peak GPU memory allocated during fine-tuning, measured in Gigabytes (GB). This is crucial for evaluating the efficiency of the unstructured backpropagation.

B.6 Implementation Details

We implement Spike-SFT in PyTorch with surrogate-gradient BPTT. In Stage 1 (SPE), we store the selected trainable coordinates as an index set (or boolean mask) and re-parameterize the trainable subset as a flat vector v. During the forward pass, the full weight tensor is reconstructed on-the-fly by scattering v into the masked positions and keeping the remaining weights frozen. Autograd therefore tracks gradients only for v, and optimizer states are allocated only for v rather than for the full dense tensor, which yields the memory savings reported in Tab. 1. The progressive parameter adjustment (PPA) schedule updates the index set between training phases based on the current weight magnitudes; when the set changes, v is refreshed from the corresponding coordinates of the current model weights.

Stage 2 (PickIt) is applied offline to saved SPE checkpoints. We load the checkpoints, form task deltas relative to the shared initialization W_0, and perform greedy fusion using the conflict-filtered merger in Eq. (8). For cross-domain fusion, we additionally align parameters before merging (Sec. 3.3) and then apply spike-statistics calibration (Eq. (10)) to stabilize membrane potential scales. The final output is a single merged checkpoint for deployment, with no additional inference-time modules.

Tab. 6 explores the sensitivity of SPE to the selection threshold θ_t, which governs the percentage of trainable parameters. Smaller thresholds select fewer parameters and reduce memory but may slightly reduce accuracy due to insufficient adaptation capacity. Larger thresholds increase trainable parameters and memory, while accuracy gains saturate, suggesting diminishing returns from activating less informative coordinates. The medium threshold used in our main experiments provides a strong accuracy–efficiency trade-off.

B.7 PickIt Cross-Domain Sensitivity to α

Table 7: PickIt Cross-Domain Sensitivity to Interpolation Coefficient α (Clipart + Painting).

Interpolation α	Clipart Acc. (%)	Painting Acc. (%)	Average Acc. (%)
0.0 (Model B only)	48.90	**78.55**	63.73
0.2	72.10	77.00	74.55
0.5	**80.50**	76.15	**78.33**
0.8	84.00	65.30	74.65
1.0 (Model A only)	**85.20**	45.10	65.15

Tab. 7 investigates the sensitivity of cross-domain fusion performance to the interpolation coefficient α. When $\alpha = 0.0$ or $\alpha = 1.0$, the performance reflects that of the individual domain-specific SPE-tuned models on the two domains (high on their own training domain, low on the other). Intermediate values of α show improved accuracy on the "other" domain, balancing the performance. The highest average accuracy is achieved at $\alpha = 0.5$, demonstrating that combining the aligned weight spaces effectively consolidates the distinct domain knowledge captured by the individual SPE-tuned models.

C Additional Experimental Results and Implementation Analysis

We provide supplementary analyses of training efficiency, model fusion, dense prediction, and spiking activity to complement the main results.

Aggregate accuracy–efficiency trade-off. Across 24 backbone–dataset combinations, mean accuracies are 88.32% for SPE, 88.41% for FFT, 87.94% for SNN LoRA, and 87.79% for SNN Adapter. Averaging the within-backbone relative savings over six backbones, SPE reduced training time by 41.9% and peak memory by 50.6% versus FFT, 32.1% and 16.4% versus SNN LoRA, and 27.1% and 27.4% versus SNN Adapter. In the implementation schedule, the initial mask is constructed once, PPA occurs once, truncated SVD is periodic, and weight scatter runs each forward pass.

C.1 Peak GPU memory and training-cost attribution

For a representative SEW-ResNet setting, FFT and SPE were profiled under matched conditions. Table 8 separates persistent parameter-side storage, BPTT state, and backward workspaces. Of the 5.140 GB reduction in peak allocated GPU memory, 0.110 GB comes from persistent parameter-side

Table 8: Matched peak-allocated GPU memory breakdown (GB). The categories describe measured memory rather than theoretical bounds.

Component	FFT	SPE
Dense weights and model buffers	0.046	0.047
Persistent gradients	0.045	0.003
AdamW moments	0.090	0.005
Active indices and PPA metadata	0.000	0.016
Persistent parameter-side subtotal	0.181	0.071
BPTT membrane and spike states	3.020	3.000
Weight-reconstruction temporaries	0.000	0.170
Saved tensors and backward workspaces	7.279	2.099
Peak allocated memory	**10.480**	**5.340**

storage. Most savings arise from saved tensors and backward workspaces, while BPTT membrane and spike states remain near 3 GB. Table 1 reports full-run wall-clock time and peak allocated GPU memory.

C.2 Comparison with existing weight-space mergers

Table 9 presents matched-pool comparisons with standard training-free mergers, using the same SPE expert pools without additional fusion-time training. Dashes denote unreported results. PickIt

Table 9: Fusion accuracy (%) using matched SPE expert pools. DomainNet columns report average accuracies over the named domains.

Merger	CIFAR-10	Clip.+Paint.	Real+Sketch	All four
Best individual SPE	95.14	–	–	–
Average of top five SPE	95.17	–	–	–
Uniform Soup	95.19	75.40	76.20	74.60
Task Arithmetic	95.22	76.00	76.80	75.30
ZipIt	95.24	–	–	75.60
Git Re-Basin	95.26	–	–	75.80
TIES	95.27	77.10	77.60	76.10
DARE+TIES	95.30	77.50	78.00	76.50
PickIt	**95.35**	**78.33**	**78.83**	**77.59**

improves on DARE+TIES by 0.05 percentage points on CIFAR-10 and by 0.83, 0.83, and 1.09 points in the three cross-domain settings, respectively.

C.3 Dense prediction and spiking activity

We further evaluate COCO dense prediction using a pretrained spiking backbone in a Mask R-CNN pipeline with identical heads. Table 10 summarizes the results. We also measure firing activity

Table 10: COCO detection and instance segmentation. Costs are relative to FFT in this setting.

Method	Box AP	Mask AP	Trainable	Time	Peak memory
FFT	41.6	36.8	100%	1.00×	1.00×
SNN LoRA	40.9	36.1	∼6%	0.91×	0.89×
SPE	**41.3**	**36.5**	∼5%	**0.82**×	**0.85**×

on SEW-ResNet/CIFAR-10 with unchanged architecture and simulation steps: SPE retains the original inference architecture; its SOP proxy is 1.04× that of FFT in this setting. Inference energy additionally depends on the resulting spike activity and deployment hardware.

Table 11: Firing activity and proxy synaptic operations (SOPs), normalized to FFT.

Method	Mean firing rate	Relative SOPs
FFT	0.121	1.00
SNN LoRA	0.118	0.98
SPE	0.124	1.04

C.4 Representative repeated-run results and selection scope

Three-seed accuracy (mean $\pm$ standard deviation) is summarized for two adaptation settings: Under

Table 12: Accuracy (%, mean $\pm$ SD over three seeds).

Setting	FFT	SNN LoRA	SPE
SEW-ResNet / CIFAR-10	95.21 ± 0.07	94.84 ± 0.09	95.17 ± 0.05
Spikformer v2 / CIFAR-100	86.18 ± 0.08	85.57 ± 0.11	86.07 ± 0.06

the matched parameter budget on SEW-ResNet/CIFAR-10, bottom-magnitude selection attains 95.19% accuracy, versus 92.56% for largest-magnitude selection and 90.13% for random selection. The three-seed results above summarize adaptation-run variability for the two reported settings.

C.5 Fusion Implementation Details

Sign election and sparse supports. Equation (8) uses sign-consistent averaging with $C_j = M_j \odot \mathbf{1}[s \neq 0, \mathrm{sign}(\Delta_j) = s]$. Weighted sign votes elect s, and only matching updates contribute to both the weighted numerator and denominator. Tied votes or empty eligible supports yield zero-valued updates. For two positive-weight experts, $\kappa = \lceil 2/2 \rceil = 1$ makes M_{cons} the union of supports; sign filtering resolves opposing updates. The expert pool $\mathcal{S}$ contains positive-weight experts, including the candidate under test. If fewer than τd coordinates of a delta are nonzero, top-τ trimming may retain all its nonzero values. With a single active expert, PickIt returns its untrimmed delta, giving the $\alpha = 0, 1$ endpoints in Table 7.

Permutation-compatible alignment. The whole-network permutation Π relabels adjacent channels and associated neuron parameters consistently. Relative to the common base W_0, the aligned delta is $\Delta_{B'} = \Pi(W_0 + \Delta_B) - W_0 = [\Pi W_0 - W_0] + \Pi\Delta_B$, which can have dense support. For compatible feedforward layers, $W'_\ell = \Pi_\ell W_\ell \Pi_{\ell-1}^\top$ and $b'_\ell = \Pi_\ell b_\ell$; the same channel permutation applies to thresholds, reset parameters, and recurrent states, with consistent permutations across residual branches.

Spike-statistics calibration. Equation (10) matches the variance of the linear pre-activation surrogate $\alpha X_A + (1-\alpha)X_{B'}$. After the nonlinear sparse-delta merge in Eq. (8), β provides approximate moment matching. For compatible affine pre-activations $X = Wx + b$, fixed calibration factors can be absorbed offline through $(W, b) \mapsto (\beta W, \beta b)$, with corresponding handling of neuron dynamics and reset states. This retains the single checkpoint and avoids an added inference module. Near-zero surrogate variances are clamped to stabilize the scale computation.

D Discussion

D.1 Limitations

Our primary evaluation focuses on classification and spike-driven backbones, with an additional COCO dense-prediction experiment; fusion assumes identical architectures and compatible checkpoints. PPA controls simultaneous trainability, not the union of modified coordinates; alignment may increase the support of deltas relative to the original initialization. Conflict-filtered merging excludes opposing-sign magnitudes, while calibration remains a heuristic rather than a guarantee for nonlinear SNN dynamics. We report training time and memory, not deployment energy. Fusion comparisons use fixed expert pools and do not estimate across-pool variability.

Potential societal impacts. Reducing SNN adaptation resources may lower barriers to reuse of pretrained models, but the measured 4% SOP proxy increase for SPE relative to FFT shows that lower training cost does not imply lower inference energy. Actual hardware energy and deployment impacts remain unmeasured.

D.2 Future Work

Building upon the Spike-SFT framework, several avenues for future research emerge. We plan to conduct a more rigorous theoretical analysis into the role and behavior of small-magnitude parameters in spike-driven SNNs to solidify the foundation of the SPE stage. Exploring alternative or dynamic criteria for parameter selection beyond simple magnitude, potentially incorporating gradient information or network connectivity analysis, could further enhance SPE's effectiveness. We also aim to apply the Spike-SFT framework to a wider range of SNN tasks and architectures, including recurrent SNNs and SNNs for graph-structured data, and investigate its applicability in resource-constrained deployment scenarios. Further development of the PickIt fusion stage could involve exploring more sophisticated SNN-specific merging or averaging techniques that explicitly account for spike dynamics, or developing methods to fuse models with minor architectural variations.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the paper's scope as efficient adaptation and weight-space fusion for pre-trained SNNs, and the claims are reflected in the proposed two-stage framework, SPE and PickIt. The experimental section evaluates the same claims through accuracy–cost comparisons, intra-domain fusion, cross-domain fusion, and ablation studies.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper includes a dedicated limitations discussion. It states that the primary evaluation focuses on classification and spike-driven backbones, with an additional COCO dense-prediction experiment, that PickIt assumes identical architectures and compatible checkpoints, and that efficiency is measured using GPU time and memory rather than direct on-device energy.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper presents a conditional LIF perturbation bound rather than a general theorem on synaptic silence; the calibration expression is a moment-matching heuristic, not a formal guarantee for nonlinear merging.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [No]

   Justification: The manuscript specifies the main algorithmic steps and evaluation datasets, but exact training hyperparameters, calibration settings, and execution commands are not supplied, limiting independent reproduction.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The paper cites public datasets and describes the principal experimental protocol, but executable code, full training commands, and scripts for all experiments are not included in this submission.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [No]

   Justification: Datasets, architectures, and broad training procedures are described, but the precise learning rates, batch sizes, seeds, and calibration protocol are not all documented.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: The Appendix reports mean and standard deviation over three seeds for two representative adaptation settings. Most fusion and efficiency results remain point estimates without independent model-pool uncertainty, so we do not claim statistical significance for them.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: The paper reports training time and peak allocated GPU memory; full per-run hardware specifications, GPU-to-result mappings, and aggregate compute budgets are not included.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work uses standard public vision and event-based benchmarks and proposes an efficient SNN adaptation and model-fusion method. We are not aware of any deviation from the NeurIPS Code of Ethics, and the camera-ready manuscript lists the accepted author team.

   Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The appendix Discussion paragraph on potential societal impacts describes resource-efficiency benefits and the possibility of increased deployment spike activity and energy; hardware energy remains unmeasured.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release high-risk assets such as generative models, scraped personal datasets, or models designed for surveillance or other sensitive applications. Therefore, special release safeguards are not applicable.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [No]

    Justification: The paper credits datasets, architectures, and prior methods, but does not comprehensively enumerate asset licenses and terms of use.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: No new dataset, code artifact, or model checkpoint is distributed as part of this manuscript.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing, user studies, data annotation by human participants, or other human-subject experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve research with human subjects, so IRB approval or an equivalent institutional review is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core methodology does not involve LLMs as an important, original, or non-standard component. Any use of LLMs for routine writing or editing, if applicable, does not affect the scientific contribution or experimental results.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.