DYNAMICS-PROJECTED ELIGIBILITY TRACES FOR ONLINE TRAINING OF SPIKING NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Spiking neural networks (SNNs) are a promising route to low-power event-driven computing, but their training typically relies on backpropagation through time (BPTT), whose activation memory scales linearly with the number of time steps. Online learning rules avoid this growth by keeping only the current-step computational graph and optionally maintaining fixed-size eligibility traces (ETs) online to approximate weight sensitivities. Existing compact presynaptic ETs use fixed decay or derive it from presynaptic dynamics, although weight sensitivities evolve according to the postsynaptic temporal Jacobian, which varies across layers and training stages. In this work, we propose Dynamics-Projected Eligibility Traces (DPET), which derives trace decay by projecting this Jacobian onto one least-squares coefficient per sample and layer. This scalar projection preserves a single presynaptic ET while adapting its decay to the postsynaptic dynamics. A state-space formulation extends DPET to neurons with richer internal states. In addition, to complement the missing feedback from future time steps in online learning, we introduce an auxiliary objective that uses later-step predictions from completed minibatches as supervision. Experiments across image, event-based, point-cloud, and language classification show gains over existing online methods with substantially lower training memory compared to BPTT. DPET provides a practical solution for memory-constrained SNN training.

1 INTRODUCTION

Spiking neural networks (SNNs) are the third generation of neural network models characterized by spiking neuronal dynamics (Maass, 1997). Their neurons communicate through sparse binary spikes rather than the dense analog activations in conventional artificial neural networks (ANNs). This event-driven computation enables low-power inference on neuromorphic chips (Akopyan et al., 2015; Davies et al., 2018; Pei et al., 2019; Yao et al., 2024b), making SNNs attractive for edge applications. To train deep SNNs directly, their dynamics are discretized into T steps so that the models can be viewed as recurrent neural networks with binary activations (Fang et al., 2021b; Eshraghian et al., 2023). Backpropagation through time (BPTT) with surrogate gradient (SG) then computes parameter gradients end-to-end (Neftci et al., 2019; Wu et al., 2018; Shrestha & Orchard, 2018). This framework supports SNN training on graphics processing units (GPUs) and has produced SNNs with competitive performance (Yao et al., 2024a; Zhou et al., 2026).

Standard BPTT, however, stores intermediate activations across time for the backward pass, so activation memory grows linearly with T (Figure 1(a)). To reduce this memory cost, online learning rules have been proposed (Bellec et al., 2020; Wang et al., 2026), as shown in Figure 1(b). These methods compute instantaneous gradients at each time step without propagating the gradients through time. Each step requires only the current-step computational graph, which can be discarded immediately after backward computation. Activation memory is therefore independent of T.

The memory efficiency comes at a cost. The temporal truncation discards historical contributions to the Jacobian of the current neuronal states with respect to their incoming weights, which we refer to as the local eligibility. Eligibility traces (ETs) can maintain this quantity online, but exact local eligibility requires per-synapse traces whose storage and update costs scale with the number of weights, limiting their use in deep SNNs (Bellec et al., 2020; Bohnstingl et al., 2023). This gives rise to the first challenge in Figure 1(c): faithfully tracking local eligibility at low storage and update

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.

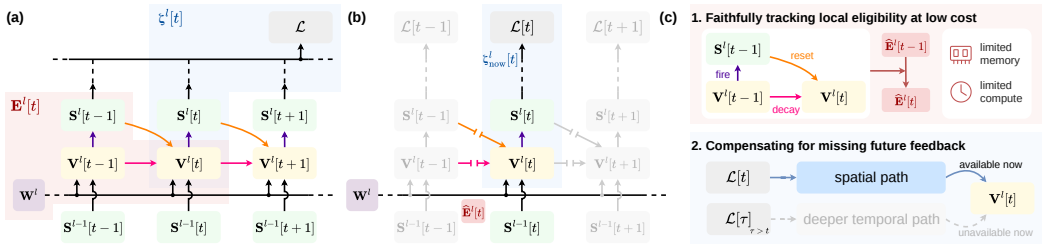


Figure 1: (a) BPTT's computational graph. (b) Online learning's computational graph. (c) Two challenges: faithfully tracking local eligibility at low cost, and compensating for missing future feedback. See Sections 3 and 4 for notation and definitions.

cost. Existing methods handle this differently. Spatial Learning Through Time (SLTT) ignores the within-layer temporal gradient path (Meng et al., 2023), Online Training Through Time (OTTT) uses fixed ET decay (Xiao et al., 2022), and Neuronal Dynamics-based Online Training (NDOT) derives ET decay from presynaptic dynamics (Jiang et al., 2024). Yet, local eligibility evolves according to the postsynaptic temporal Jacobian, which varies across layers and over training. A second, distinct challenge remains even with exact local eligibility: since each online backward pass is driven only by the current loss, it omits the feedback that future losses would send through deeper neuronal states. Improving the eligibility approximation alone cannot recover this missing future feedback.

To address the first challenge, we introduce Dynamics-Projected Eligibility Traces (DPET), which derive ET decay from a least-squares scalar projection of the postsynaptic temporal Jacobian. For leaky integrate-and-fire (LIF) neurons, this reduces the neuron-wise temporal factors to one state-dependent coefficient per sample and layer, yielding an ET that matches the shape of presynaptic inputs with storage independent of T. We further extend DPET to neurons with richer internal states through a state-space formulation in Appendix A.4. Gradient diagnostics show that DPET approximates local eligibility more faithfully than fixed-decay tracing. As the online update omits feedback from future time steps, we introduce lagged temporal distillation, which supervises current outputs with later-step predictions from completed minibatches without reading ahead within the current sequence. We also introduce membrane potential regularization to stabilize learning when surrogate gradients are weak during early training. Across image, event-based, point-cloud, and language benchmarks, our method performs better than existing online rules, and approaches full BPTT in several settings. Our contributions can be summarized as follows:

(1) We propose DPET, which derives compact presynaptic ET decay from postsynaptic temporal Jacobians, and extend its formulation to multi-state neurons.

(2) We introduce lagged temporal distillation and membrane potential regularization as auxiliary objectives to improve online learning performance.

(3) Our methods provide competitive or improved performance compared with existing online rules across diverse tasks, while requiring substantially less training memory than full BPTT.

2 Related Work

BPTT with SG and its refinements. Deep SNNs are typically simulated on T time steps so that they can be trained end-to-end as recurrent neural networks using BPTT (Werbos, 1990; Wu et al., 2018; Shrestha & Orchard, 2018), with surrogate gradients addressing the non-differentiability of the spike firing function (Neftci et al., 2019). This framework enables the training of deep spiking residual networks (Fang et al., 2021a; Hu et al., 2025) and spiking Transformers (Zhou et al., 2023; Yao et al., 2023; Zhou et al., 2024; Fang et al., 2025). Refined neuron models (Fang et al., 2021b; Yao et al., 2022; Huang et al., 2024), per-step objectives (Deng et al., 2022), and SNN-tailored variants (Kim & Panda, 2021; Duan et al., 2022) of batch normalization (BN) (Ioffe & Szegedy, 2015) further improve training performance. Despite these advances, BPTT stores intermediate activations across T steps, causing activation memory to grow linearly with T. For instance, training a Spike-element-wise ResNet-34 (Fang et al., 2021a) with $T = 4$ requires roughly $7\times$ the peak memory of its ANN counterpart (Huang et al., 2026).

Memory-efficient SNN training. To reduce the memory footprint of SNN training, prior work has explored several alternatives to standard BPTT. Gradient checkpointing lowers activation storage by discarding intermediate activations during the forward pass and recomputing them during the backward pass (Singh et al., 2022; Bencheikh et al., 2024; Huang et al., 2026). It preserves exact BPTT gradients but introduces extra computation. Reversible SNNs reconstruct intermediate activations in reverse during the backward pass, reducing storage at the cost of strict architectural constraints and reconstruction overhead (Zhang & Zhang, 2024; Hu et al., 2024). BPTT-to-BP methods replace temporal backpropagation with equilibrium- or rate-based approximations, reducing memory and computation at the cost of applicability to tasks with temporal dependencies (Xiao et al., 2021; Wu et al., 2023; Kheradpisheh et al., 2022; Yu et al., 2024). Online learning instead restricts backpropagation to the current time step, making activation memory independent of T without recomputing activations. However, temporal truncation introduces errors in temporal credit assignment.

Online learning. Online learning methods differ in how they handle temporal dependencies during gradient computation. SLTT discards temporal gradient paths (Meng et al., 2023). Under the same truncation, Online Spiking Renormalization (OSR) improves normalization consistency (Zhu et al., 2024), while the Hybrid-Driven Leaky Integrate-and-Fire (HD-LIF) model modifies neuronal dynamics for gradient coherence. Note that HD-LIF emits real-valued outputs rather than the binary spikes considered here (Hao et al., 2026). Forward Propagation Through Time (FPTT) takes a different approach by incorporating history through a dynamically regularized instantaneous objective (Kag & Saligrama, 2021; Yin et al., 2023). Other methods propagate parameter sensitivities forward in time. Real-Time Recurrent Learning (RTRL) maintains exact parameter sensitivities (Williams & Zipser, 1989), while factorized and sparse variants reduce that cost (Mujika et al., 2018; Menick et al., 2021). For SNNs, eligibility propagation (e-prop) and Online Spatio-Temporal Learning (OSTL) maintain local eligibility with per-synapse ETs (Bellec et al., 2020; Bohnstingl et al., 2023), requiring each sample to carry a trace over all synapses. To reduce trace storage, OTTT uses fixed-decay presynaptic ETs (Xiao et al., 2022), while NDOT derives decay from presynaptic dynamics (Jiang et al., 2024). BrainTrace's pp-prop factorizes eligibility into pre- and postsynaptic traces (Wang et al., 2026). DPET instead uses a least-squares scalar projection of the postsynaptic temporal Jacobian to set the decay of a single input-shaped ET per sample and layer.

A separate limitation concerns downstream error feedback, since current-step spatial backpropagation omits feedback from future losses through deeper neuronal states (Bellec et al., 2020). Online Training with Postsynaptic Estimates (OTPE) approximates these downstream temporal effects by tracking parameter influence on subsequent-layer activity (Summe et al., 2024). At the supervision level, temporal knowledge sharing transfers predictions across steps (Dong et al., 2024), but its targets can depend on later outputs of the current sequence. Our lagged teacher instead uses later-step predictions from completed minibatches, providing auxiliary supervision without lookahead.

3 PRELIMINARIES

3.1 FORWARD PASS OF SNNS

For an L-layer SNN with soft-reset leaky integrate-and-fire (LIF) neurons (Gerstner et al., 2014), the discrete-time dynamics are given by Equation (1).[1] Unless stated otherwise, the derivations in this work use this neuron model. Indices $l \in \{1, \dots, L\}$ and $t \in \{1, \dots, T\}$ run over layers and time steps, and n_l is the width of layer l. The linear map g^l produces the input current $\mathbf{X}^l[t] \in \mathbb{R}^{n_l}$ from the previous layer's output using weights $\mathbf{W}^l \in \mathbb{R}^{n_l \times n_{l-1}}$. The membrane potential before spike emission is $\mathbf{V}^l[t] \in \mathbb{R}^{n_l}$, with decay factor $\lambda \in (0, 1)$ and firing threshold $V_{\text{th}} > 0$; the Heaviside function Θ emits a spike once $\mathbf{V}^l[t] \geq V_{\text{th}}$, after which soft reset subtracts V_{th} from the potential. Neuronal updates are element-wise, and resting and reset potentials are omitted for simplicity. Outputs $\mathbf{S}^l[t] \in \{0, 1\}^{n_l}$ are binary while the network input $\mathbf{S}^0[t]$ may be real-valued (Rathi & Roy, 2023).

$$\begin{aligned} \mathbf{X}^l[t] &= g^l(\mathbf{S}^{l-1}[t]; \mathbf{W}^l), \\ \mathbf{V}^l[t] &= \lambda(\mathbf{V}^l[t-1] - V_{\text{th}}\mathbf{S}^l[t-1]) + \mathbf{X}^l[t], \\ \mathbf{S}^l[t] &= \Theta(\mathbf{V}^l[t] - V_{\text{th}}). \end{aligned} \tag{1}$$

[1] Bold symbols denote tensors; their nonbold, indexed counterparts denote scalar elements.

The readout weight $\mathbf{W}^o \in \mathbb{R}^{C\times n_L}$ projects the final-layer spikes to the output $\mathbf{O}[t] \in \mathbb{R}^C$, where C is the output size. The loss is defined either on the averaged output or separately at each time step:

$$\mathbf{O}[t] = \mathbf{W}^o \mathbf{S}^L[t], \qquad \mathcal{L}_{\text{offline}} = \ell\left(\frac{1}{T}\sum_{t=1}^{T}\mathbf{O}[t], y\right), \qquad \mathcal{L}_{\text{online}} = \frac{1}{T}\sum_{t=1}^{T}\ell(\mathbf{O}[t], y). \tag{2}$$

Here ℓ is a loss function (e.g., cross-entropy) and y is the target. In the online formulation, each instantaneous loss $\mathcal{L}[t] = \ell(\mathbf{O}[t], y)/T$ depends only on the current output, so its gradient can be computed without accessing any future outputs.

3.2 BPTT WITH SG

To optimize these objectives, BPTT propagates error signals backward through both layers and time in the unfolded graph (Werbos, 1990). Since Θ has zero derivative almost everywhere and is non-differentiable at the threshold, we replace its derivative with a surrogate σ' while keeping the forward spike function unchanged (Neftci et al., 2019; Wu et al., 2018; Shrestha & Orchard, 2018). The one-step temporal Jacobian is

$$\mathbf{\Phi}^l[t] = \frac{\mathrm{d}\mathbf{V}^l[t]}{\mathrm{d}\mathbf{V}^l[t-1]} = \lambda\left[\mathbf{I} - V_{\text{th}}\,\mathrm{diag}\big(\sigma'(\mathbf{V}^l[t-1] - V_{\text{th}})\big)\right]. \tag{3}$$

Here $\mathbf{\Phi}^l[t] \in \mathbb{R}^{n_l\times n_l}$ is diagonal because neuronal updates are element-wise. Let $\mathcal{L}$ denote either objective in Equation (2). Grouping the gradient paths by the time step at which they leave layer l gives the following decomposition of the weight gradient:

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\mathbf{W}^l} = \sum_{t=1}^{T} \boldsymbol{\zeta}^l[t]\,\mathbf{E}^l[t], \qquad \boldsymbol{\zeta}^l[t] \triangleq \frac{\partial\mathcal{L}}{\partial\mathbf{V}^l[t]}. \tag{4}$$

The downstream error signal $\boldsymbol{\zeta}^l[t] \in \mathbb{R}^{n_l}$ (blue) carries credit from $\mathbf{V}^l[t]$ to every loss it influences after leaving the layer; it retains temporal paths through deeper layers but excludes the recurrence within layer l. This within-layer recurrence is captured by the local eligibility $\mathbf{E}^l[t]$ (red):

$$\mathbf{E}^l[t] \triangleq \frac{\mathrm{d}\mathbf{V}^l[t]}{\mathrm{d}\mathbf{W}^l} = \sum_{\tau=1}^{t}\left(\prod_{s=\tau+1}^{t}\mathbf{\Phi}^l[s]\right) g'^{l}_{\mathbf{W}}(\mathbf{S}^{l-1}[\tau]). \tag{5}$$

Here $g'^{l}_{\mathbf{W}}(\mathbf{S}) \triangleq \partial g^l(\mathbf{S};\mathbf{W}^l)/\partial\mathbf{W}^l$ is linear in $\mathbf{S}$. An empty product equals $\mathbf{I}$. We call $\mathbf{E}^l[t]$ **local eligibility** because its temporal paths are confined to layer l; downstream paths are carried by $\boldsymbol{\zeta}^l[t]$. An **eligibility trace (ET)** is an online state that maintains or approximates this local eligibility. Appendix A.1 proves this decomposition and relates it to reverse-mode BPTT.

4 METHOD

The decomposition in Section 3.2 identifies two requirements for online learning: a compact representation of local eligibility and an error signal available at the current step.

4.1 CHALLENGE 1. COMPACT LOCAL ELIGIBILITY

The exact local eligibility $\mathbf{E}^l[t]$ in Equation (5) can be updated online (Bohnstingl et al., 2023):

$$\mathbf{E}^l[t] = \mathbf{\Phi}^l[t]\mathbf{E}^l[t-1] + g'^{l}_{\mathbf{W}}(\mathbf{S}^{l-1}[t]), \qquad \mathbf{E}^l[0] = \mathbf{0}. \tag{6}$$

For one sample in a dense layer, $\mathbf{E}^l[t]$ has shape $n_l \times n_l \times n_{l-1}$, but only $n_l n_{l-1}$ entries can be nonzero because each neuron's state depends only on its incoming weights. Maintaining these entries still requires $\mathcal{O}(n_l n_{l-1})$ storage and computation per update, which is costly for deep SNNs. The first challenge is therefore to **track local eligibility with low storage and update costs**.

To obtain a compact ET, we approximate each temporal Jacobian by a scalar shared across postsynaptic neurons, $\mathbf{\Phi}^l[s] \approx \phi^l[s]\mathbf{I}$. Since $g'^{l}_{\mathbf{W}}$ is linear in its input, the scalar coefficients in Equation (5) can be absorbed into the presynaptic activity:

$$\mathbf{E}^l[t] \approx \sum_{\tau=1}^{t}\left(\prod_{s=\tau+1}^{t}\phi^l[s]\right) g'^{l}_{\mathbf{W}}(\mathbf{S}^{l-1}[\tau]) = g'^{l}_{\mathbf{W}}\Big(\textstyle\sum_{\tau=1}^{t}\big(\prod_{s=\tau+1}^{t}\phi^l[s]\big)\mathbf{S}^{l-1}[\tau]\Big). \tag{7}$$

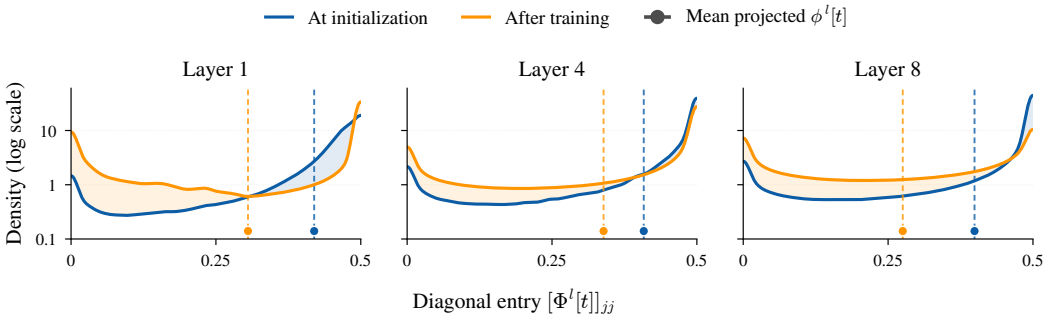


Figure 2: Distributions of the diagonal entries of $\mathbf{\Phi}^l[t]$ across layers and training stages in Spiking VGG11 on CIFAR-100 ($\lambda = 0.5$). Dashed lines mark the scalar projection $\phi^l[t] = \mathrm{tr}(\mathbf{\Phi}^l[t])/n_l$. See Appendix C.4 for details.

Define the sum inside $g'^l_{\mathbf{W}}$ as the compact trace $\widehat{\mathbf{E}}^l[t] \in \mathbb{R}^{n_{l-1}}$, giving $\mathbf{E}^l[t] \approx g'^l_{\mathbf{W}}(\widehat{\mathbf{E}}^l[t])$. It can be updated online as

$$\widehat{\mathbf{E}}^l[t] = \phi^l[t]\widehat{\mathbf{E}}^l[t-1] + \mathbf{S}^{l-1}[t], \qquad \widehat{\mathbf{E}}^l[0] = \mathbf{0}. \tag{8}$$

The remaining choice is how to compute $\phi^l[t]$ from $\mathbf{\Phi}^l[t]$. Minimizing $\|\mathbf{\Phi}^l[t] - \phi^l[t]\mathbf{I}\|_F^2$ gives the mean of the diagonal entries:

$$\phi^l[t] = \frac{\mathrm{tr}(\mathbf{\Phi}^l[t])}{n_l} = \lambda\left[1 - V_{\mathrm{th}}\,\mathrm{avg}\big(\sigma'(\mathbf{V}^l[t-1] - V_{\mathrm{th}})\big)\right]. \tag{9}$$

Here avg averages over the n_l neurons within this layer for each sample. We call this method **Dynamics-Projected Eligibility Traces (DPET)**. The resulting trace stores n_{l-1} values per sample and costs $\mathcal{O}(n_l + n_{l-1})$ per update, both independent of T. Appendix A.3 explains why a shared input trace requires a scalar decay, Appendix A.4 extends DPET to multi-state neurons, and Appendix A.6 analyzes the layer-wise reduction error.

Relation to prior methods. Setting $\phi^l[t] = 0$ or λ recovers the SLTT and OTTT eligibility approximations, respectively (Meng et al., 2023; Xiao et al., 2022). NDOT instead derives its coefficient from presynaptic dynamics, whereas local eligibility follows the postsynaptic recurrence (Jiang et al., 2024) (detailed in Appendix A.7). Figure 2 visualizes the neuronal coefficients and their scalar projection. These coefficients vary across layers and drift during training, and DPET's scalar projection follows these shifts. Table A1 summarizes these rules.

4.2 Challenge 2. Missing Future Feedback

For BPTT applied to $\mathcal{L}_{\mathrm{online}}$, the downstream error signal in Equation (4) contains current and future contributions (Figure 1(c); Appendix A.2):

$$\boldsymbol{\zeta}^l[t] = \boldsymbol{\zeta}^l_{\mathrm{now}}[t] + \boldsymbol{\zeta}^l_{\mathrm{future}}[t], \qquad \boldsymbol{\zeta}^l_{\mathrm{now}}[t] \triangleq \frac{\partial \mathcal{L}[t]}{\partial \mathbf{V}^l[t]}. \tag{10}$$

$\boldsymbol{\zeta}^l_{\mathrm{now}}[t]$ is obtained by spatial backpropagation of $\mathcal{L}[t]$ with temporal dependencies detached. The future term carries feedback from later losses through deeper neuronal states and is unavailable at step t. SLTT, OTTT, NDOT, and DPET therefore estimate gradients using only the current error signal (Meng et al., 2023; Xiao et al., 2022; Jiang et al., 2024):

$$\widehat{\nabla_{\mathbf{W}^l}\mathcal{L}}[t] = \boldsymbol{\zeta}^l_{\mathrm{now}}[t]\, g'^l_{\mathbf{W}}(\widehat{\mathbf{E}}^l[t]). \tag{11}$$

Improving the eligibility approximation alone cannot recover this missing future feedback. The second challenge is therefore to **compensate for missing future feedback without reading ahead**.

For classification, we introduce **lagged temporal distillation (LTD)**, which supervises current predictions with the network's later-step predictions on previously processed samples of the same class. This extends temporal self-distillation (Dong et al., 2024). For C classes, the teacher bank stores a probability vector $\mathbf{b}_{y,t} \in \mathbb{R}^C$ for each class y and supervised step $t < T$. After completing a batch

of N samples with label distributions $\boldsymbol{\pi}_i \in [0,1]^C$ (i indexes samples in the batch), we form teacher candidates from weighted averages of detached later-step logits and update the bank:

$$\widetilde{\mathbf{p}}_{i,t} = \mathrm{softmax}\left(\frac{\sum_{s=t+1}^{T} \gamma^{s-t-1}\mathbf{O}_i[s]}{\xi \sum_{s=t+1}^{T} \gamma^{s-t-1}}\right), \qquad \mathbf{b}_{y,t} \leftarrow \eta \mathbf{b}_{y,t} + (1-\eta)\frac{\sum_{i=1}^{N} \pi_{i,y}\widetilde{\mathbf{p}}_{i,t}}{\sum_{i=1}^{N} \pi_{i,y}}. \tag{12}$$

Here ξ is the temperature, γ discounts more distant steps, and η is the bank momentum. Entries with $\sum_i \pi_{i,y} = 0$ are unchanged. For each new batch, we form the teacher $\mathbf{q}_{i,t}$ and final target $\mathbf{p}_{i,t}$ using only the bank and reliabilities $r_{y,t} \in [0,1]$ accumulated from previous batches:

$$\mathbf{q}_{i,t} = \sum_y \pi_{i,y}\mathbf{b}_{y,t}, \qquad \bar{r}_{i,t} = \sum_y \pi_{i,y} r_{y,t}, \qquad \mathbf{p}_{i,t} = (1-\bar{r}_{i,t})\boldsymbol{\pi}_i + \bar{r}_{i,t}\mathbf{q}_{i,t}. \tag{13}$$

Reliability downweights entries with few observations or high prediction variance and is zero for unseen entries (see Appendix B.1). We add the batch-averaged distillation loss

$$\mathcal{L}_{\mathrm{KD}}[t] = \frac{\omega \xi^2}{N} \sum_{i=1}^{N} \bar{r}_{i,t}\, \mathrm{KL}(\mathbf{p}_{i,t} \,\|\, \mathrm{softmax}(\mathbf{O}_i[t]/\xi)). \tag{14}$$

Here ω is the loss weight, and $\mathcal{L}_{\mathrm{KD}}[T] = 0$. Teacher quantities are detached, so gradients flow only through current-step predictions. Although LTD does not reconstruct the missing $\boldsymbol{\zeta}^l_{\mathrm{future}}[t]$, it provides auxiliary supervision that complements DPET.

4.3 Membrane Potential Regularization

When membrane potentials remain far below or above the firing threshold, neurons stay silent or fire at every step. At these potentials, the sigmoid surrogate derivative used in most experiments, $\sigma'(x) = 4e^{-4x}/(1+e^{-4x})^2$ with $x = V - V_{\mathrm{th}}$, is small. Gradients from the task and distillation losses can therefore become too weak to correct these firing states. To strengthen the optimization signal in such weak-gradient regimes, we apply a warm-up penalty inspired by membrane potential regularization (Guo et al., 2022; 2023) to a set $\mathcal{I}_{\mathrm{reg}}$ of layers using DPET:

$$R_{\mathrm{mem}}[t] = \frac{1}{|\mathcal{I}_{\mathrm{reg}}|} \sum_{l \in \mathcal{I}_{\mathrm{reg}}} \mathrm{avg}_{i,j}\left[\left(\frac{|V^l_{i,j}[t] - V_{\mathrm{th}}|}{|V_{\mathrm{th}}|} - m_v\right)_+^2\right], \quad \mathcal{L}_{\mathrm{mem}}[t] = c_{\mathrm{mem}} R_{\mathrm{mem}}[t]. \tag{15}$$

Here $(z)_+ = \max(z, 0)$, $\mathrm{avg}_{i,j}$ averages over batch samples and neurons in each layer, and m_v is the threshold-relative margin. The penalty is zero within the margin and pushes more distant potentials toward the threshold. The regularizer's weight c_{mem} is adaptively adjusted using a moving average of R_{mem} and annealed to zero by the end of warm-up; Appendix B.3 gives the exact schedule.

4.4 Implementation Details

BN compatibility. Consider the commonly used $\mathrm{Conv} \to \mathrm{BN} \to \mathrm{LIF}$ block in SNNs. The substitution in Equation (7) requires $g'^l_{\mathbf{W}}$ to be linear in its input, whereas the weight Jacobian of the combined Conv–BN transform is generally nonlinear in the input during training. DPET blocks therefore either move BN after the neuron ($\mathrm{Conv} \to \mathrm{LIF}_{\mathrm{DPET}} \to \mathrm{BN}$) or omit BN and apply scaled weight standardization (sWS) to the projection weights (Brock et al., 2021). Appendix B.2 provides the compatibility analysis and normalization details.

Selective DPET placement. DPET can be applied to all compatible blocks or only a subset. We denote the indices of the layers using DPET by $\mathcal{I}_{\mathrm{DPET}}$, with $\mathcal{I}_{\mathrm{reg}} \subseteq \mathcal{I}_{\mathrm{DPET}}$. Selective placement limits trace storage and the number of blocks requiring BN relocation or removal; unselected layers use SLTT with their original BN placement. The selected layer indices are fixed for each architecture and listed in Appendix C.2.

Overall objective. The per-step objective combines the task loss and two auxiliary losses:

$$\mathcal{L}_{\mathrm{total}}[t] = w[t]\left[\frac{1}{N}\sum_{i=1}^{N} \ell(\mathbf{O}_i[t], y_i) + \mathcal{L}_{\mathrm{KD}}[t]\right] + \mathcal{L}_{\mathrm{mem}}[t]. \tag{16}$$

Table 1: Representative SNN results. Architectures and protocols vary; shaded rows use DPET, and starred rows are local best-checkpoint results.

Dataset	Method	Online	Architecture	Norm.	T	Accuracy(%)
CIFAR-10	TET (Deng et al., 2022)	No	ResNet-19	BN†	6	$94.50_{\pm 0.07}$
	TEBN (Duan et al., 2022)	No	ResNet-19	TEBN	6	95.60
	SLTT (Meng et al., 2023)	Yes	ResNet-18	BN	6	$94.44_{\pm 0.21}$
	OSR+OTS (Zhu et al., 2024)	Yes	VGGSNN	OSR+OTS	4	94.35
	OTTT_{A} (Xiao et al., 2022)	Yes	VGG11	sWS	6	$93.52_{\pm 0.06}$
	NDOT_{A} (Jiang et al., 2024)	Yes	VGG11	sWS	6	94.90
	pp-prop* (Wang et al., 2026)	Yes	VGG11	BN‡	6	94.56
	DPET*	Yes	VGG11	BN‡	6	$94.89_{\pm 0.02}$
	DPET + both objectives*	Yes	VGG11	BN‡	6	$95.24_{\pm 0.09}$
CIFAR-100	TET (Deng et al., 2022)	No	ResNet-19	BN†	6	$74.72_{\pm 0.28}$
	TEBN (Duan et al., 2022)	No	ResNet-19	TEBN	6	78.76
	BPTT* (Fang et al., 2025)	No	MaxResNet-18	BN†	4	$82.68_{\pm 0.06}$
	SLTT (Meng et al., 2023)	Yes	ResNet-18	BN	6	$74.38_{\pm 0.30}$
	OSR+OTS (Zhu et al., 2024)	Yes	VGGSNN	OSR+OTS	4	76.48
	OTTT_{A} (Xiao et al., 2022)	Yes	VGG11	sWS	6	$71.05_{\pm 0.04}$
	NDOT_{A} (Jiang et al., 2024)	Yes	VGG11	sWS	6	76.47
	pp-prop* (Wang et al., 2026)	Yes	VGG11	BN‡	6	74.37
	DPET*	Yes	VGG11	BN‡	6	$76.97_{\pm 0.09}$
	DPET + both objectives*	Yes	VGG11	BN‡	6	$77.98_{\pm 0.10}$
	Selective DPET*	Yes	MaxResNet-18	BN+sWS	4	$81.21_{\pm 0.16}$
	Selective DPET + both objectives*	Yes	MaxResNet-18	BN+sWS	4	$82.08_{\pm 0.35}$
CIFAR10-DVS	TET (Deng et al., 2022)	No	VGGSNN	BN†	10	$83.17_{\pm 0.15}$
	TEBN (Duan et al., 2022)	No	VGGSNN	TEBN	10	84.90
	SLTT (Meng et al., 2023)	Yes	VGG11	BN	10	$82.20_{\pm 0.95}$
	OSR+OTS (Zhu et al., 2024)	Yes	VGGSNN	OSR+OTS	10	82.40
	OTTT_{A} (Xiao et al., 2022)	Yes	VGG11	sWS	10	$76.27_{\pm 0.05}$
	NDOT_{A} (Jiang et al., 2024)	Yes	VGG11	sWS	10	77.40
	pp-prop* (Wang et al., 2026)	Yes	VGG11	BN‡	10	82.80
	DPET*	Yes	VGG11	BN‡	10	$84.13_{\pm 0.35}$
	DPET + both objectives*	Yes	VGG11	BN‡	10	$85.97_{\pm 0.81}$
TinyImageNet	LTL (Yang et al., 2022)	No	VGG13 (LIF)	n/r	16	55.37
	BPTT (Huang et al., 2024)	No	VGG13 (CLIF)	BN†	6	64.13
	BPTT*	No	VGG13 (LIF/CLIF)	BN†	6	65.82/65.46
	SLTT*	Yes	VGG13 (LIF/CLIF)	BN	6	63.44/63.94
	DPET*	Yes	VGG13 (LIF/CLIF)	sWS	6	64.16/64.46
	Selective DPET*	Yes	VGG13 (LIF/CLIF)	BN+sWS	6	64.85/64.97

*: our implementation; BN†: BN after folding T into the batch dimension; BN‡: post-neuron BN; mean $\pm$ std: on three random seeds; entries without std are single results; both objectives: membrane regularization + LTD.

Here y_i is the target for sample i, and $w[t] \geq 0$ weights the task and distillation losses. With both auxiliary losses disabled and $w[t] = 1/T$, $\sum_t \mathcal{L}_{\text{total}}[t]$ equals the batch mean of $\mathcal{L}_{\text{online}}$. Spatial backpropagation through the current-step graph gives the error signal $\partial \mathcal{L}_{\text{total}}[t]/\partial \mathbf{V}_i^l[t]$ for each sample. For layers in $\mathcal{I}_{\text{DPET}}$, this signal replaces $\boldsymbol{\zeta}_{\text{now}}^l[t]$ in Equation (11), with sample-wise weight-gradient contributions summed over the batch; other layers use ordinary spatial gradients.

Training procedure. All T forward steps still advance neuronal states and traces. At each step, we calculate and accumulate the current-step gradients. After every step, temporal dependencies are detached and the current computation graph is discarded. Parameters remain fixed within the sequence and are updated once after step T. Consequently, activation and trace storage remain independent of T. LTD additionally requires T-dependent storage for its teacher bank and cached logits, which is typically much smaller than activation and trace storage across all network layers (see Figure 3 and Appendix B.1). Algorithm 1 in Appendix B.5 gives the complete training procedure. Appendix B.4 describes an optional strategy that performs backward passes at K of the T steps to reduce training time cost.

5 Experiments

We evaluate task performance, component contributions, gradient fidelity, and training efficiency in this section. Experimental settings and additional results are provided in Appendices C and D.

Table 2: ScanObjectNN test accuracy (%, $T = 4$; three-seed mean±std).

Method	Spiking PointNet	SDT
BPTT	67.72±0.34	86.54±0.56
SLTT	67.23±0.40	86.07±0.20
OTTT_A	67.70±0.18	85.91±0.26
NDOT_A	68.04±0.30	85.63±0.48
pp-prop	66.17±0.61	85.05±0.32
DPET	68.08±0.62	85.93±0.47
+ both obj.	67.99±0.21	86.23±0.61

Table 3: SpikingLM fine-tuning accuracy (%, $T = 4$; three-seed mean±std).

Method	SST-2	MRPC	QNLI	RTE
BPTT	89.40±0.29	80.23±0.50	83.64±0.24	58.86±1.10
SLTT	88.57±0.07	79.25±0.51	82.54±0.28	57.40±0.63
OTTT_A	89.45±0.13	79.41±0.25	83.89±0.18	58.12±0.71
NDOT_A	88.61±0.20	80.88±0.44	83.26±0.35	59.37±0.89
DPET	89.11±0.40	81.13±0.56	84.07±0.21	59.21±1.10
+ both obj.†	89.47±0.13	80.93±1.35	84.39±0.58	59.40±0.75

†: LTD + Q/K/V spike-rate penalty (Appendix D.8).

5.1 Is DPET Effective Across Tasks and Models?

Image and event-based classification. Table 1 summarizes results on four image and event-based classification tasks. On CIFAR-10 and CIFAR-100, DPET achieves 94.89% and 76.97% accuracy, respectively; incorporating LTD and membrane regularization further improves the results to 95.24% and 77.98%. With MaxResNet-18 (Fang et al., 2025), selective DPET (DPET/SLTT hybrid) with both auxiliary objectives reaches 82.08% on CIFAR-100, approaching BPTT's 82.68%. On CIFAR10-DVS, the full method achieves 85.97% accuracy, surpassing other online rules. On TinyImageNet, selective DPET outperforms SLTT for both LIF (64.85% vs. 63.44%) and two-state complementary LIF (CLIF) neurons (64.97% vs. 63.94%), showing that DPET extends effectively to multi-state neuron dynamics. ImageNet results are provided in Appendix D.1.

Point-cloud classification. We further evaluate DPET on ScanObjectNN using two spiking point-cloud backbones (Table 2). On Spiking PointNet (Ren et al., 2023), DPET achieves 68.08% test accuracy, outperforming BPTT and SLTT and slightly exceeding NDOT_A's 68.04%. On Spiking Discrepancy Transformer (SDT) (Lu et al., 2026), DPET with both auxiliary objectives achieves 86.23%, remaining competitive with BPTT (86.54%) and SLTT (86.07%), demonstrating its applicability across different point-cloud architectures.

Language understanding. We fine-tune SpikingLM BERT (Liang et al., 2026) on four GLUE tasks (Table 3), applying the evaluated trace-based method to the Q/K/V projections while using SLTT in the remaining layers. Here, membrane regularization is replaced by a spike-rate based penalty, since its threshold-relative band saturates on the pretrained model's signed Q/K/V potentials (Appendix D.8). DPET-based conditions achieve the highest mean accuracy among the compared online methods on all four GLUE tasks, with the largest improvement on QNLI (84.39% for DPET with both auxiliary objectives vs. 83.89% for OTTT_A). These results demonstrate that DPET extends effectively to fine-tuning pretrained spiking language models.

5.2 Do the Auxiliary Objectives Improve Task Performance?

To assess the contributions of LTD and membrane regularization, we conduct ablation studies on CIFAR-100 and CIFAR10-DVS (Table 4). Each objective improves accuracy over DPET, while their combination yields 1.01% gain on CIFAR-100 and 1.84% gain on CIFAR10-DVS. The combined objectives also improve accuracy by 0.35% on CIFAR-10 with VGG11 and by 0.87% on CIFAR-100 with MaxResNet-18 (Table 1).

Table 4: Ablations on CIFAR-100 and CIFAR10-DVS (accuracy, %; three-seed mean±std).

Condition	CIFAR-100	CIFAR10-DVS
DPET	76.97±0.09	84.13±0.35
+ mem. reg.	77.24±0.13	85.47±0.21
+ LTD	77.34±0.40	85.33±0.47
+ both objectives	77.98±0.10	85.97±0.81

5.3 Does DPET Preserve Temporal Credit More Faithfully?

To assess gradient fidelity, we compare five estimators on three SLTT-trained VGG11 checkpoints using the same 64 CIFAR-100 images ($T = 6$), with matched weights, forward trajectories, and objectives. The exact local-gradient reference combines current-step spatial error signals with exact

Table 5: Relative L_2 error on Spiking VGG11 for CIFAR-100 ($T = 6$; three-seed mean±std).

	Exact local gradient	Full BPTT gradient
SLTT	$0.3462_{\pm 0.0023}$	$0.4795_{\pm 0.0029}$
OTTT_A	$0.2866_{\pm 0.0070}$	$0.3578_{\pm 0.0013}$
NDOT_A	$0.3532_{\pm 0.0065}$	$0.3889_{\pm 0.0019}$
pp-prop	$0.1281_{\pm 0.0009}$	$0.4247_{\pm 0.0028}$
DPET	$0.1540_{\pm 0.0013}$	$0.3419_{\pm 0.0023}$

Table 6: Sequential CIFAR-100 Accuracy (%, three-seed mean±std).

Method	Accuracy
BPTT+TET	$57.86_{\pm 0.27}$
SLTT	$54.62_{\pm 0.40}$
OTTT_A	$56.04_{\pm 0.81}$
NDOT_A	$52.03_{\pm 0.85}$
pp-prop	$48.93_{\pm 1.44}$
DPET	$57.28_{\pm 0.36}$
+ LTD	$57.50_{\pm 0.49}$

Table 7: Update time (ms) / peak allocated memory (GiB).

Spiking VGG (RTX 3090, b128, bf16)				
Method	$T = 4$	8	16	32
BPTT	115/3.28	237/6.37	481/12.55	OOM
SLTT	114/0.98	226/0.98	456/0.98	909/0.99
OTTT_A	129/1.05	258/1.05	508/1.05	1014/1.04
DPET (torch)	128/1.05	251/1.05	497/1.05	988/1.05
DPET (Triton)	**104/0.93**	**200/0.93**	**393/0.93**	**781/0.93**
SDT (RTX 5090, b32, fp32)				
Method	$T = 4$	8	16	32
BPTT	430/29.28	OOM	OOM	OOM
SLTT	405/9.10	780/9.10	1529/9.10	3028/9.10
OTTT_A	419/9.13	794/9.13	1545/9.13	3049/9.13
DPET (torch)	418/9.13	794/9.13	1545/9.13	3050/9.13
DPET (Triton)	**333/8.43**	**622/8.43**	**1200/8.43**	**2358/8.43**

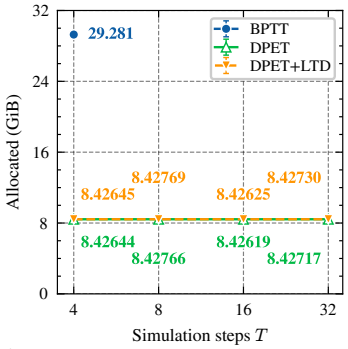


Figure 3: SDT peak memory; BPTT OOMs for $T \geq 8$.

within-layer eligibility, while full BPTT additionally includes future feedback through deeper layers. The detailed experiment protocol is provided in Appendix C.5. DPET achieves the lowest mean relative L_2 error against the full BPTT gradient, although its error against the local reference is higher than that of pp-prop (Table 5). A paired analysis of DPET and DPET+LTD at checkpoints with frozen historical teachers is given in Appendix D.6.

To complement the gradient error analysis with a task-level diagnostic, we evaluate the online rules on Sequential CIFAR-100 ($T = 32$), where optimization relies strongly on temporal credit assignment. On this task, DPET+LTD achieves the highest accuracy of 57.50% among the compared online methods (Table 6). Together with the direct gradient measurements, these results illustrate the impact of temporal-gradient approximation quality on optimization.

5.4 What are the Memory and Computation Costs?

We implement DPET with both PyTorch and Triton backends. The Triton backend uses two fused kernels: one for the neuron forward pass and one for the surrogate-gradient backward pass. For both Spiking VGG and SDT, DPET's peak memory stays nearly constant as T grows, while BPTT runs out of memory at longer horizons (Table 7). Triton further reduces both update time and peak memory on VGG and SDT, cutting SDT update time by 20–23% relative to PyTorch. LTD adds less than 0.14 MiB to DPET's peak allocation in SDT (Figure 3).

6 Conclusion

In this work, we presented Dynamics-Projected Eligibility Traces (DPET) for memory-efficient online training of SNNs. DPET projects the postsynaptic temporal Jacobian onto a sample- and layer-dependent scalar, allowing historical local eligibility to be retained in a single input-shaped trace whose storage is independent of the number of time steps. To address the complementary loss of future feedback, we introduced lagged temporal distillation, together with membrane potential regularization and compatible trace placement for practical training. Across image, event-based, point-cloud, and language tasks, DPET delivers strong performance across diverse architectures while substantially reducing training memory cost compared with BPTT. A discussion of limitations and future work is provided in Appendix E.

REPRODUCIBILITY STATEMENT

We will provide anonymized code as supplementary material for review and release it on GitHub upon acceptance.

AI USE STATEMENT

We use AI tools to assist with manuscript refining, code refactoring, data analysis and visualization. The authors developed the conceptual framework and derivations, conducted the experiments, and drafted the manuscript. AI tools were not used to generate research ideas.

REFERENCES

Filipp Akopyan, Jun Sawada, Andrew Cassidy, Rodrigo Alvarez-Icaza, John Arthur, Paul Merolla, Nabil Imam, Yutaka Nakamura, Pallab Datta, Gi-Joon Nam, Brian Taba, Michael Beakes, Bernard Brezzo, Jente B. Kuang, Rajit Manohar, William P. Risk, Bryan Jackson, and Dharmendra S. Modha. Truenorth: Design and tool flow of a 65 mW 1 million neuron programmable neurosynaptic chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(10):1537–1557, 2015.

Guillaume Bellec, Franz Scherr, Anand Subramoney, Elias Hajek, Darjan Salaj, Robert Legenstein, and Wolfgang Maass. A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature Communications*, 11(1):3625, 2020.

Wadjih Bencheikh, Jan Finkbeiner, and Emre Neftci. Optimal gradient checkpointing for sparse and recurrent architectures using off-chip memory. *arXiv preprint arXiv:2412.11810*, 2024.

Thomas Bohnstingl, Stanisław Woźniak, Angeliki Pantazi, and Evangelos Eleftheriou. Online spatio-temporal learning in deep neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 34(11):8894–8908, 2023.

Andrew Brock, Soham De, and Samuel L Smith. Characterizing signal propagation to close the performance gap in unnormalized resnets. In *International Conference on Learning Representations*, 2021.

Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 113–123, 2019.

Ekin Dogus Cubuk, Barret Zoph, Jon Shlens, and Quoc V. Le. RandAugment: Practical automated data augmentation with a reduced search space. In *Advances in Neural Information Processing Systems*, volume 33, pp. 18613–18624, 2020.

Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, Yuyun Liao, Chit-Kwan Lin, Andrew Lines, Ruokun Liu, Deepak Mathaikutty, Steven McCoy, Arnab Paul, Jonathan Tse, Guruguhanathan Venkataramanan, Yi-Hsin Weng, Andreas Wild, Yoonseok Yang, and Hong Wang. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

Shikuang Deng, Yuhang Li, Shanghang Zhang, and Shi Gu. Temporal efficient training of spiking neural network via gradient re-weighting. In *The Tenth International Conference on Learning Representations*, 2022.

Terrance DeVries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

Yiting Dong, Dongcheng Zhao, and Yi Zeng. Temporal knowledge sharing enable spiking neural network learning from past and future. *IEEE Transactions on Artificial Intelligence*, 5(7):3524–3534, 2024.

Chaoteng Duan, Jianhao Ding, Shiyan Chen, Zhaofei Yu, and Tiejun Huang. Temporal effective batch normalization in spiking neural networks. In *Advances in Neural Information Processing Systems*, volume 35, pp. 34377–34390, 2022.

Jason K. Eshraghian, Max Ward, Emre O. Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D. Lu. Training spiking neural networks using lessons from deep learning. *Proceedings of the IEEE*, 111(9):1016–1054, 2023.

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*, volume 34, pp. 21056–21069, 2021a.

Wei Fang, Zhaofei Yu, Yanqi Chen, Timothée Masquelier, Tiejun Huang, and Yonghong Tian. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2661–2671, 2021b.

Yuetong Fang, Deming Zhou, Ziqing Wang, Hongwei Ren, Zecui Zeng, Lusong Li, Shibo Zhou, and Renjing Xu. Spiking neural networks need high-frequency information. *Advances in Neural Information Processing Systems*, 38:103404–103425, 2025.

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.

Yufei Guo, Xinyi Tong, Yuanpei Chen, Liwen Zhang, Xiaode Liu, Zhe Ma, and Xuhui Huang. Recdis-snn: Rectifying membrane potential distribution for directly training spiking neural networks. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 326–335. IEEE, 2022.

Yufei Guo, Xiaode Liu, Yuanpei Chen, Liwen Zhang, Weihang Peng, Yuhan Zhang, Xuhui Huang, and Zhe Ma. RMP-Loss: Regularizing membrane potential distribution for spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 17391–17401, 2023.

Zecheng Hao, Yifan Huang, Zijie Xu, Wenxuan Liu, Yuanhong Tang, Zhaofei Yu, and Tiejun Huang. Rethinking SNN online training and deployment: Gradient-coherent learning via hybrid-driven LIF model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20202–20211, 2026.

Jiakui Hu, Man Yao, Xuerui Qiu, Yuhong Chou, Yuxuan Cai, Ning Qiao, Yonghong Tian, Bo Xu, and Guoqi Li. High-performance temporal reversible spiking neural networks with $\mathcal{O}(L)$ training memory and $\mathcal{O}(1)$ inference cost. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 19516–19530, 2024.

Yifan Hu, Lei Deng, Yujie Wu, Man Yao, and Guoqi Li. Advancing spiking neural networks toward deep residual learning. *IEEE Transactions on Neural Networks and Learning Systems*, 36(2): 2353–2367, 2025.

Yifan Huang, Wei Fang, Zecheng Hao, Zhengyu Ma, and Yonghong Tian. Towards lossless memory-efficient training of spiking neural networks via gradient checkpointing and spike compression. In *The Fourteenth International Conference on Learning Representations*, 2026.

Yulong Huang, Xiaopeng Lin, Hongwei Ren, Haotian Fu, Yue Zhou, Zunchang Liu, Biao Pan, and Bojun Cheng. CLIF: Complementary leaky integrate-and-fire neuron for spiking neural networks. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 19949–19972. PMLR, 2024.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pp. 448–456, 2015.

Haiyan Jiang, Giulia De Masi, Huan Xiong, and Bin Gu. NDOT: Neuronal dynamics-based online training for spiking neural networks. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 21806–21823, 2024.

Anil Kag and Venkatesh Saligrama. Training recurrent neural networks via forward propagation through time. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pp. 5189–5200, 2021.

Saeed Reza Kheradpisheh, Maryam Mirsadeghi, and Timothée Masquelier. Spiking neural networks trained via proxy. *IEEE Access*, 10:70769–70778, 2022.

Youngeun Kim and Priyadarshini Panda. Revisiting batch normalization for training low-latency deep spiking neural networks from scratch. *Frontiers in Neuroscience*, 15:773954, 2021.

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

Ya Le and Xuan Yang. Tiny ImageNet visual recognition challenge. Cs231n project report, Stanford University, 2015. URL `https://cs231n.stanford.edu/reports/2015/pdfs/yle_project.pdf`.

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.

Yuhang Li, Youngeun Kim, Hyoungseob Park, Tamar Geller, and Priyadarshini Panda. Neuromorphic data augmentation for training spiking neural networks. In *European Conference on Computer Vision*, pp. 631–649, 2022.

Yu Liang, Zijian Zhou, Wenjie Wei, Shuai Wang, Honglin Cao, Ammar Belatreche, Yu Yang, Malu Zhang, Yang Yang, and Haizhou Li. SpikingLM: Towards fully spiking language model. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *The Seventh International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Yijie Lu, Zhiyi Pan, Renrui Zhang, Yanhao Jia, Ronggang Wang, and Zhaokun Zhou. Spiking discrepancy transformer for point cloud analysis. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=7Brnh0aNFn.

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

Qingyan Meng, Mingqing Xiao, Shen Yan, Yisen Wang, Zhouchen Lin, and Zhi-Quan Luo. Towards memory- and time-efficient backpropagation for training spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6166–6176, 2023.

Jacob Menick, Erich Elsen, Utku Evci, Simon Osindero, Karen Simonyan, and Alex Graves. Practical real time recurrent learning with a sparse approximation. In *The Ninth International Conference on Learning Representations*, 2021.

Asier Mujika, Florian Meier, and Angelika Steger. Approximating real-time recurrent learning with random kronecker factors. In *Advances in Neural Information Processing Systems*, volume 31, pp. 6594–6603, 2018.

Samuel G. Müller and Frank Hutter. Trivialaugment: Tuning-free yet state-of-the-art data augmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 774–782, 2021.

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.

Jing Pei, Lei Deng, Sen Song, Mingguo Zhao, Youhui Zhang, Shuang Wu, Guanrui Wang, Zhe Zou, Zhenzhi Wu, Wei He, et al. Towards artificial general intelligence with hybrid tianjic chip architecture. *Nature*, 572(7767):106–111, 2019.

Nitin Rathi and Kaushik Roy. Diet-snn: A low-latency spiking neural network with direct input encoding and leakage and threshold optimization. *IEEE Transactions on Neural Networks and Learning Systems*, 34(6):3174–3182, 2023.

Dayong Ren, Zhe Ma, Yuanpei Chen, Weihang Peng, Xiaode Liu, Yuhan Zhang, and Yufei Guo. Spiking pointnet: Spiking neural networks for point clouds. In *Advances in Neural Information Processing Systems*, 2023.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115 (3):211–252, 2015.

Xinyu Shi and Zhaofei Yu. Training deep spiking neural networks without normalization. In *International Conference on Machine Learning*, 2026.

Sumit Bam Shrestha and Garrick Orchard. Slayer: Spike layer error reassignment in time. In *Advances in Neural Information Processing Systems*, volume 31, pp. 1419–1428, 2018.

Sonali Singh, Anup Sarma, Sen Lu, Abhronil Sengupta, Mahmut T Kandemir, Emre Neftci, Vijaykrishnan Narayanan, and Chita R Das. Skipper: Enabling efficient snn training through activation-checkpointing and time-skipping. In *2022 55th IEEE/ACM International Symposium on Microarchitecture*, pp. 565–581. IEEE, 2022.

Thomas M. Summe, Clemens J. S. Schaefer, and Siddharth Joshi. Estimating post-synaptic effects for online training of feed-forward SNNs. In *Proceedings of the International Conference on Neuromorphic Systems*, pp. 264–271, 2024.

Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Duc Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1588–1597, 2019.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In

Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, pp. 353–355, 2018.

Chaoming Wang, Xingsi Dong, Zilong Ji, Mingqing Xiao, Jiedong Jiang, Xiao Liu, Yuxiang Huan, and Si Wu. Model-agnostic linear-memory online learning in spiking neural networks. *Nature Communications*, 17:1745, 2026.

Paul J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

Ronald J. Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989.

Jibin Wu, Yansong Chua, Malu Zhang, Guoqi Li, Haizhou Li, and Kay Chen Tan. A tandem learning rule for effective training and rapid inference of deep spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 34(1):446–460, 2023.

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.

Mingqing Xiao, Qingyan Meng, Zongpeng Zhang, Yisen Wang, and Zhouchen Lin. Training feedback spiking neural networks by implicit differentiation on the equilibrium state. *Advances in neural information processing systems*, 34:14516–14528, 2021.

Mingqing Xiao, Qingyan Meng, Zongpeng Zhang, Di He, and Zhouchen Lin. Online training through time for spiking neural networks. In *Advances in Neural Information Processing Systems*, volume 35, pp. 20717–20730, 2022.

Qu Yang, Jibin Wu, Malu Zhang, Yansong Chua, Xinchao Wang, and Haizhou Li. Training spiking neural networks with local tandem learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 12662–12676, 2022.

Man Yao, JiaKui Hu, Zhaokun Zhou, Li Yuan, Yonghong Tian, Bo Xu, and Guoqi Li. Spike-driven transformer. In *Advances in Neural Information Processing Systems*, volume 36, pp. 64043–64058, 2023.

Man Yao, JiaKui Hu, Tianxiang Hu, Yifan Xu, Zhaokun Zhou, Yonghong Tian, Bo XU, and Guoqi Li. Spike-driven transformer v2: Meta spiking neural network architecture inspiring the design of next-generation neuromorphic chips. In *The Twelfth International Conference on Learning Representations*, 2024a.

Man Yao, Ole Richter, Guangshe Zhao, Ning Qiao, Yannan Xing, Dingheng Wang, Tianxiang Hu, Wei Fang, Tugba Demirci, Michele De Marchi, et al. Spike-based dynamic computing with asynchronous sensing-computing neuromorphic chip. *Nature communications*, 15(1):4464, 2024b.

Xingting Yao, Fanrong Li, Zitao Mo, and Jian Cheng. Glif: A unified gated leaky integrate-and-fire neuron for spiking neural networks. In *Advances in Neural Information Processing Systems*, volume 35, pp. 32160–32171, 2022.

Bojian Yin, Federico Corradi, and Sander M. Bohté. Accurate online training of dynamical spiking neural networks through forward propagation through time. *Nature Machine Intelligence*, 5(5): 518–527, 2023.

Chengting Yu, Lei Liu, Gaoang Wang, Erping Li, and Aili Wang. Advancing training efficiency of deep spiking neural networks through rate-based backpropagation. In *Advances in Neural Information Processing Systems*, volume 37, pp. 115786–115815, 2024.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6023–6032, 2019.

Hong Zhang and Yu Zhang. Memory-efficient reversible spiking neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(15):16759–16767, 2024.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *The Sixth International Conference on Learning Representations*, 2018.

Hanle Zheng, Yujie Wu, Lei Deng, Yifan Hu, and Guoqi Li. Going deeper with directly-trained larger spiking neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35 (12):11062–11070, 2021.

Chenlin Zhou, Han Zhang, Zhaokun Zhou, Liutao Yu, Liwei Huang, Xiaopeng Fan, Li Yuan, Zhengyu Ma, Huihui Zhou, and Yonghong Tian. Qkformer: Hierarchical spiking transformer using q-k attention. In *Advances in Neural Information Processing Systems*, volume 37, pp. 13074–13098, 2024.

Chenlin Zhou, Liutao Yu, Zhaokun Zhou, Han Zhang, Jiaqi Wang, Huihui Zhou, Zhengyu Ma, and Yonghong Tian. Spikingformer: A key foundation model for spiking neural networks. In

Proceedings of the AAAI Conference on Artificial Intelligence, volume 40, pp. 2236–2244, 2026.

Zhaokun Zhou, Yuesheng Zhu, Chao He, Yaowei Wang, Shuicheng YAN, Yonghong Tian, and Li Yuan. Spikformer: When spiking neural network meets transformer. In *The Eleventh International Conference on Learning Representations*, 2023.

Yaoyu Zhu, Jianhao Ding, Tiejun Huang, Xiaodong Xie, and Zhaofei Yu. Online stabilization of spiking neural networks. In *The Twelfth International Conference on Learning Representations*, 2024.

A METHOD DERIVATIONS

The component derivations below use $\Phi_j^l[t] = [\mathbf{\Phi}^l[t]]_{jj}$ and, for a dense layer, $E_{jk}^l[t] = \mathrm{d}V_j^l[t]/\mathrm{d}W_{jk}^l$. For a dense layer, $[\mathbf{E}^l[t]]_{a,j,k} = \mathrm{d}V_a^l[t]/\mathrm{d}W_{jk}^l$, so $E_{jk}^l[t] = [\mathbf{E}^l[t]]_{j,j,k}$. The main-text contraction is $[\boldsymbol{\zeta}^l[t]\mathbf{E}^l[t]]_{jk} = \sum_a \zeta_a^l[t][\mathbf{E}^l[t]]_{a,j,k}$. Where matrix shapes are used below, the weight axes of $\mathbf{E}^l[t]$ and $g'^l_{\mathbf{W}}$ are flattened into one axis of length $p_l = |\mathbf{W}^l|$. The vector $\mathbf{E}_j^l[t]$ collects the incoming-weight entries for unit j; the shared input trace remains $\widehat{\mathbf{E}}^l[t]$.

A.1 FULL PROJECTION GRADIENT AND PATH DECOMPOSITION

For the basic fully connected stack in Equation (1), the chain rule separates the full membrane derivative into a spatial branch and a temporal branch (Meng et al., 2023):

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}V_j^l[t]} = \sigma'(V_j^l[t] - V_{\mathrm{th}}) \sum_m W_{mj}^{l+1} \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}V_m^{l+1}[t]} + \Phi_j^l[t+1] \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}V_j^l[t+1]}, \qquad l < L. \tag{17}$$

At $l = L$, the spatial branch is $\sigma'(V_j^L[t] - V_{\mathrm{th}}) \sum_c W_{cj}^o \, \partial\mathcal{L}/\partial O_c[t]$, where the partial derivative treats the other output nodes as independent. The temporal branch vanishes at $t = T$. Backward evaluation over layers and time gives the full membrane derivatives. Each occurrence of W_{jk}^l contributes the direct derivative $S_k^{l-1}[t]$ to $V_j^l[t]$; each occurrence of b_j^l contributes one. Summing over these occurrences gives the conventional reverse-mode form

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}W_{jk}^l} = \sum_t \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}V_j^l[t]} S_k^{l-1}[t], \qquad \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}b_j^l} = \sum_t \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}V_j^l[t]}. \tag{18}$$

The full membrane derivative here includes within-layer recurrence. Expanding that recurrence and regrouping by the time at which a path leaves layer l gives Equation (4). The partial derivative in that equation excludes the within-layer temporal paths now contained in the local eligibility $\mathbf{E}^l[t]$; it is not the full membrane derivative above.

For the feedforward layer stack in Section 3.2, a perturbation of W_{jk}^l enters $V_j^l[\tau]$ at some time τ. It can follow recurrent edges within layer l until time $t \geq \tau$, then leave through the current spatial output. Because there are no feedback connections from deeper layers to layer l, the path cannot return after leaving. Summing the path products before this departure gives $E_{jk}^l[t]$; summing the products after departure gives $\zeta_j^l[t]$. Grouping by the unique departure time proves Equation (10) for the surrogate-linearized graph. Using a full membrane adjoint in place of the cut-graph signal would count the within-layer recurrent segment again.

BPTT differentiates the weighted task loss $\mathcal{L}_{\mathrm{task}} = \sum_{t=1}^T w[t]\ell(\mathbf{O}[t], y)$, the auxiliary-free case of Equation (16), through the complete spatio-temporal computational graph. Let $\mathbf{X}_{\mathrm{proj}}^l[t] = g^l(\mathbf{A}^{l-1}[t])$ denote the affine projection output, before any intervening normalization. Define its full BPTT error as $\boldsymbol{\epsilon}^l[t] = \mathrm{d}\mathcal{L}_{\mathrm{task}}/\mathrm{d}\mathbf{X}_{\mathrm{proj}}^l[t]$. The weight gradient sums the contribution of each occurrence of the shared projection:

$$\frac{\mathrm{d}\mathcal{L}_{\mathrm{task}}}{\mathrm{d}\mathbf{W}^l} = \sum_{t=1}^T \mathcal{P}_l\big(\mathbf{A}^{l-1}[t], \boldsymbol{\epsilon}^l[t]\big)\,, \tag{19}$$

where $\mathcal{P}_l$ is the weight-gradient operator of projection g^l: an outer product for a fully connected layer and a cross-correlation for a convolution. The full projection-output gradient $\boldsymbol{\epsilon}^l[t]$ contains contributions from the current and future losses propagated through subsequent neuronal states. BPTT obtains it by storing and differentiating the unfolded graph. Exact forward-in-time propagation of all corresponding network-wide sensitivities has the cost of real-time recurrent learning (Williams & Zipser, 1989).

A.2 DOWNSTREAM PATH CONVENTION

The learning signal requires a separate path convention. Let $\mathbf{D}_{s,t}^l$ be the Jacobian of output $\mathbf{O}[s]$ with respect to $\mathbf{V}^l[t]$ in a graph that cuts the recurrent edges of layer l. This graph retains its current spatial

edges and all temporal edges in deeper layers. The cut prevents the learning signal from counting the within-layer recurrence already contained in $E^l_{jk}[t]$. Writing $\mathbf{a}[s] = w[s]\nabla_{\mathbf{O}[s]}\ell(\mathbf{O}[s], y)$, define

$$\boldsymbol{\zeta}^l[t] = \sum_{s=t}^{T} (\mathbf{D}^l_{s,t})^\top \mathbf{a}[s] = \underbrace{(\mathbf{D}^l_{t,t})^\top \mathbf{a}[t]}_{\boldsymbol{\zeta}^l_{\text{now}}[t]} + \underbrace{\sum_{s=t+1}^{T} (\mathbf{D}^l_{s,t})^\top \mathbf{a}[s]}_{\boldsymbol{\zeta}^l_{\text{future}}[t]}. \tag{20}$$

This *downstream learning signal* assigns credit to the current state through the outputs it affects after leaving layer l. Its future term contains paths through deeper neuronal states at later steps.

A.3 Why the Shared Trace Requires Scalar Decay

With a scalar temporal factor, each update keeps the trace in the same input-shaped form. We first consider one sample in a dense LIF layer with n outputs and r inputs, omitting the layer index. Let $\mathbf{a} \in \mathbb{R}^{r\times 1}$ be a projection input and $\Delta\mathbf{W} \in \mathbb{R}^{n\times r}$ a weight perturbation. Define the weight-Jacobian operator $\mathcal{G}(\mathbf{a}) : \mathbb{R}^{n\times r} \to \mathbb{R}^{n\times 1}$ by

$$\mathcal{G}(\mathbf{a})[\Delta\mathbf{W}] = \Delta\mathbf{W}\mathbf{a}. \tag{21}$$

This operator is linear in $\mathbf{a}$. It is the action of the main-text weight Jacobian: $\mathcal{G}(\mathbf{a})[\Delta\mathbf{W}] = g'_{\mathbf{W}}(\mathbf{a})\,\mathrm{vec}(\Delta\mathbf{W})$, where $g'_{\mathbf{W}}(\mathbf{a}) \in \mathbb{R}^{n\times nr}$. The exact local sensitivity operator $\mathcal{J}[t] : \mathbb{R}^{n\times r} \to \mathbb{R}^{n\times 1}$ therefore obeys

$$\mathcal{J}[t] = \mathbf{D}[t]\mathcal{J}[t-1] + \mathcal{G}(\mathbf{S}^{l-1}[t]), \qquad \mathbf{D}[t] = \mathrm{diag}(\Phi_1[t], \ldots, \Phi_n[t]) \in \mathbb{R}^{n\times n}, \tag{22}$$

with zero initial sensitivity. Here $\mathbf{S}^{l-1}[t] \in \mathbb{R}^{r\times 1}$ is the input from the basic main-text stack; compatible blocks use $\mathbf{A}^{l-1}[t]$ instead.

DPET maintains $\widehat{\mathcal{J}}[t] = \mathcal{G}(\widehat{\mathbf{E}}[t])$ with $\widehat{\mathbf{E}}[t] \in \mathbb{R}^{r\times 1}$. Absorbing a diagonal output factor $\mathbf{D}$ into a common input $\mathbf{e} \in \mathbb{R}^{r\times 1}$ requires

$$\mathbf{D}\Delta\mathbf{W}\mathbf{a} = \Delta\mathbf{W}\mathbf{e} \quad \text{for every } \Delta\mathbf{W} \in \mathbb{R}^{n\times r}. \tag{23}$$

Since the rows of $\Delta\mathbf{W}$ vary independently, row j requires $\mathbf{e} = D_{jj}\mathbf{a}$. For $\mathbf{a} \neq 0$, all diagonal entries must therefore coincide: $\mathbf{D} = \phi\mathbf{I}_n$ with $\phi \in \mathbb{R}$. When $\mathbf{a} = 0$, the historical contribution vanishes and places no restriction on ϕ. This necessity concerns a single shared input and the unchanged projection Jacobian; representations that retain output-specific factors have different constraints.

Approximating $\mathbf{D}[t]$ by $\phi[t]\mathbf{I}_n$ gives

$$\begin{aligned}\widehat{\mathcal{J}}[t] &= \phi[t]\mathcal{G}(\widehat{\mathbf{E}}[t-1]) + \mathcal{G}(\mathbf{S}^{l-1}[t]) \\ &= \mathcal{G}\Big(\phi[t]\widehat{\mathbf{E}}[t-1] + \mathbf{S}^{l-1}[t]\Big).\end{aligned} \tag{24}$$

Consequently, every historical contribution can be accumulated into one input-shaped trace. In the main-text Jacobian notation, unrolling this approximation makes the input-linearity step explicit:

$$\begin{aligned}\frac{\mathrm{d}\mathbf{V}^l[t]}{\mathrm{d}\,\mathrm{vec}(\mathbf{W}^l)^\top} &\approx \sum_{\tau=1}^{t} \left(\prod_{s=\tau+1}^{t} \phi^l[s]\right) g'^l_{\mathbf{W}}(\mathbf{S}^{l-1}[\tau]) \\ &= g'^l_{\mathbf{W}}\left(\sum_{\tau=1}^{t} \left(\prod_{s=\tau+1}^{t} \phi^l[s]\right) \mathbf{S}^{l-1}[\tau]\right) = g'^l_{\mathbf{W}}(\widehat{\mathbf{E}}^l[t]).\end{aligned} \tag{25}$$

Both sides have shape $n_l \times p_l$, where p_l is the number of projection weights. Convolutional projection Jacobians are also linear in the input, so the same common-scalar construction applies. The necessity proof above uses independent dense weight rows; special connectivity and degenerate inputs can allow other factorizations.

The single-trace representation requires a scalar temporal factor. We choose its value by minimizing the following error:

$$\phi[t] = \arg\min_{\varphi\in\mathbb{R}} \|\mathbf{D}[t] - \varphi\mathbf{I}_n\|_F^2 = \frac{1}{n}\sum_{j=1}^{n} \Phi_j[t]. \tag{26}$$

Differentiating the objective gives $2\sum_j(\varphi-\Phi_j[t])=0$. Thus input linearity motivates the common scalar, and equal-weight least squares selects the arithmetic mean. The resulting weight-eligibility storage is r values per dense sample. Per-weight eligibility stores nr values, while an input/output trace pair stores $r+n$ values before forming its outer product. These counts exclude bias traces, neuronal states, and local tangent buffers. They establish a storage comparison; actual execution time also depends on the implementation.

A.4 State-Space Eligibility Projection

The generalized derivation fixes one sample and one postsynaptic unit j in layer l. Let d_l be the number of state variables in the unit and let r_l be the number of scalar projection inputs to that unit. For a dense layer, $r_l = n_{l-1}$. For a convolution, r_l is the number of scalar inputs in one local receptive field. Weight sharing is handled by the layer operator $\mathcal{P}_l$.

The unit receives one scalar effective projection current $X_j^l[t]$. It has one scalar visible state $Y_j^l[t]$ that drives its spike. We write the local view of the main-text projection input as $\mathbf{A}_j^{l-1}[t]$:

$$\mathbf{Z}_j^l[t] = F_{j,l}(\mathbf{Z}_j^l[t-1], X_j^l[t]), \qquad U_j^l[t] = (\mathbf{w}_j^l)^\top \mathbf{A}_j^{l-1}[t] + b_j^l, \tag{27}$$

$$X_j^l[t] = \Gamma_j^l[t](U_j^l[t]), \qquad Y_j^l[t] = \psi_{j,l}(\mathbf{Z}_j^l[t]), \qquad S_j^l[t] = \rho_l(Y_j^l[t]). \tag{28}$$

Here $F_{j,l} : \mathbb{R}^{d_l} \times \mathbb{R} \to \mathbb{R}^{d_l}$ and $\psi_{j,l} : \mathbb{R}^{d_l} \to \mathbb{R}$. The map $\Gamma_j^l[t] : \mathbb{R} \to \mathbb{R}$ is the effective current transformation. It is the identity for a pure affine projection. If a stop-gradient scalar RMS limiter is enabled, it is the scalar multiplication $\Gamma_j^l[t](u) = \kappa_j^l[t]u$, where $\kappa_j^l[t] \in \mathbb{R}$ is detached from the derivative. Define $\gamma_j^l[t] \triangleq \partial\Gamma_j^l[t](u)/\partial u \in \mathbb{R}$ at $u = U_j^l[t]$; thus $\gamma_j^l[t] = 1$ for the identity map and $\gamma_j^l[t] = \kappa_j^l[t]$ for the detached limiter. The quantities $\mathbf{Z}_j^l[t] \in \mathbb{R}^{d_l\times 1}$, $\mathbf{A}_j^{l-1}[t] \in \mathbb{R}^{r_l\times 1}$, and $\mathbf{w}_j^l \in \mathbb{R}^{r_l\times 1}$ are vectors. The scalars $U_j^l[t]$, $X_j^l[t]$, and $Y_j^l[t]$ have shape $\mathbb{R}$. The spike has shape $S_j^l[t] \in \{0,1\}$. The local visible Jacobian is $\mathbf{C}_j^l[t] \triangleq \partial Y_j^l[t]/\partial \mathbf{Z}_j^l[t] \in \mathbb{R}^{1\times d_l}$. The vector $\mathbf{A}_j^{l-1}[t]$ is the local view of the main-text projection input $\mathbf{A}^{l-1}[t]$. The forward map $\rho_l : \mathbb{R} \to \{0,1\}$ is the Heaviside spike function; its derivative is replaced by the same surrogate derivative σ' used in the main text.

The subscripted $\mathbf{A}_j^l[t]$ below denotes a state-transition Jacobian. The superscripted $\mathbf{A}^{l-1}[t]$ denotes the main-text projection input.

Define the local Jacobians and the exact parameter sensitivity by

$$\mathbf{A}_j^l[t] \triangleq \frac{\partial \mathbf{Z}_j^l[t]}{\partial \mathbf{Z}_j^l[t-1]} \in \mathbb{R}^{d_l\times d_l}, \qquad \mathbf{B}_j^l[t] \triangleq \frac{\partial \mathbf{Z}_j^l[t]}{\partial X_j^l[t]} \in \mathbb{R}^{d_l\times 1}, \tag{29}$$

$$\mathbf{Q}_j^l[t] \triangleq \frac{\partial X_j^l[t]}{\partial \mathbf{w}_j^l} \in \mathbb{R}^{r_l\times 1}, \qquad \mathbf{P}_j^l[t] \triangleq \frac{\mathrm{d}\mathbf{Z}_j^l[t]}{\mathrm{d}(\mathbf{w}_j^l)^\top} \in \mathbb{R}^{d_l\times r_l}. \tag{30}$$

By the scalar chain rule, $\mathbf{Q}_j^l[t] = \gamma_j^l[t]\mathbf{A}_j^{l-1}[t]$. Thus the identity current map gives $\mathbf{Q}_j^l[t] = \mathbf{A}_j^{l-1}[t]$, while the stop-gradient scalar RMS limiter gives $\mathbf{Q}_j^l[t] = \kappa_j^l[t]\mathbf{A}_j^{l-1}[t]$. The local derivative holds the upstream input sequence $\{\mathbf{A}_j^{l-1}[t]\}_{t=1}^T$ and the initial state fixed. The exact forward sensitivity follows from the chain rule:

$$\mathbf{P}_j^l[t] = \mathbf{A}_j^l[t]\mathbf{P}_j^l[t-1] + \mathbf{B}_j^l[t](\mathbf{Q}_j^l[t])^\top, \qquad \mathbf{P}_j^l[0] = \mathbf{0}_{d_l\times r_l}. \tag{31}$$

The dimensions verify each term: $\mathbf{AP} \in \mathbb{R}^{d_l\times r_l}$ and $\mathbf{BQ}^\top \in \mathbb{R}^{d_l\times r_l}$. We define the ordered product by $\prod_{s=\tau+1}^t \mathbf{A}_j^l[s] \triangleq \mathbf{A}_j^l[t]\cdots\mathbf{A}_j^l[\tau+1]$. Unrolling Equation (31) gives

$$\mathbf{P}_j^l[t] = \sum_{\tau=1}^t \left(\prod_{s=\tau+1}^t \mathbf{A}_j^l[s]\right)\mathbf{B}_j^l[\tau](\mathbf{Q}_j^l[\tau])^\top \in \mathbb{R}^{d_l\times r_l}, \tag{32}$$

where an empty product is $\mathbf{I}_{d_l\times d_l}$. Each direct injection $\mathbf{B}_j^l[\tau](\mathbf{Q}_j^l[\tau])^\top$ has rank at most one in the state and weight axes. The sum can contain several state directions after the transition Jacobians act on different historical input tangents.

The one-trace approximation keeps one state direction and one coefficient column:

$$\widehat{\mathbf{P}}_j^l[t] \triangleq \mathbf{B}_j^l[t](\widehat{\mathbf{E}}_j^l[t])^\top, \qquad \widehat{\mathbf{E}}_j^l[t] \in \mathbb{R}^{r_l \times 1}. \tag{33}$$

Equation (33) is an approximation for $d_l > 1$. It is exact at the first input injection when $\mathbf{P}_j^l[0] = 0$. In the implemented one-state LIF case, $\mathbf{C}_j^l[t]\mathbf{B}_j^l[t] = 1$ and the representation remains exact at every step. More generally, when $d_l = 1$ and $\mathbf{C}_j^l[s]\mathbf{B}_j^l[s] \neq 0$ for each step $s \leq t$, the scalar recurrence is exact by induction. For a multi-state neuron, it assumes that the accumulated parameter sensitivity remains well represented by the current input-induced state direction. For $t = 1$, the historical term is absent and $\widehat{\mathbf{E}}_j^l[1] = \mathbf{Q}_j^l[1]$; the visible matching coefficient is needed from $t = 2$ onward.

For $t \geq 2$ and a small weight perturbation $\Delta\mathbf{w}_j^l \in \mathbb{R}^{r_l \times 1}$, the scalar $(\widehat{\mathbf{E}}_j^l[t-1])^\top \Delta\mathbf{w}_j^l \in \mathbb{R}$ gives its coefficient. The product $\mathbf{B}_j^l[t-1]((\widehat{\mathbf{E}}_j^l[t-1])^\top \Delta\mathbf{w}_j^l) \in \mathbb{R}^{d_l \times 1}$ gives the approximated state displacement. Define the discarded residual by

$$\mathbf{R}_j^l[t-1] \triangleq \mathbf{P}_j^l[t-1] - \mathbf{B}_j^l[t-1](\widehat{\mathbf{E}}_j^l[t-1])^\top \in \mathbb{R}^{d_l \times r_l}. \tag{34}$$

The one-trace method stores $\widehat{\mathbf{E}}$ and discards $\mathbf{R}$.

Projecting the exact recurrence onto the visible state gives

$$\mathbf{C}_j^l[t]\mathbf{P}_j^l[t] = \mathbf{C}_j^l[t]\mathbf{A}_j^l[t]\mathbf{P}_j^l[t-1] + \mathbf{C}_j^l[t]\mathbf{B}_j^l[t](\mathbf{Q}_j^l[t])^\top \in \mathbb{R}^{1 \times r_l}. \tag{35}$$

For $t \geq 2$, substituting the residual decomposition into the projected recurrence gives

$$\begin{aligned}\mathbf{C}_j^l[t]\mathbf{P}_j^l[t] = &\left(\mathbf{C}_j^l[t]\mathbf{A}_j^l[t]\mathbf{B}_j^l[t-1]\right)(\widehat{\mathbf{E}}_j^l[t-1])^\top + \mathbf{C}_j^l[t]\mathbf{A}_j^l[t]\mathbf{R}_j^l[t-1] \\ &+ \left(\mathbf{C}_j^l[t]\mathbf{B}_j^l[t]\right)(\mathbf{Q}_j^l[t])^\top \in \mathbb{R}^{1 \times r_l}.\end{aligned} \tag{36}$$

The one-trace approximation drops the residual term $\mathbf{C}_j^l[t]\mathbf{A}_j^l[t]\mathbf{R}_j^l[t-1]$. The visible matching rule controls the first historical term. The residual remains explicit.

The numerator in the visible matching coefficient is a scalar. Its factors have shapes $\mathbf{C} \in \mathbb{R}^{1 \times d_l}$, $\mathbf{A} \in \mathbb{R}^{d_l \times d_l}$, and $\mathbf{B} \in \mathbb{R}^{d_l \times 1}$. For $t \geq 2$, we choose $\phi_j^l[t] \in \mathbb{R}$ to match the transported previous tangent to the current tangent after the visible-state projection:

$$\phi_j^l[t] = \frac{\mathbf{C}_j^l[t]\mathbf{A}_j^l[t]\mathbf{B}_j^l[t-1]}{\mathbf{C}_j^l[t]\mathbf{B}_j^l[t]}, \qquad \mathbf{C}_j^l[t]\mathbf{B}_j^l[t] \neq 0. \tag{37}$$

This choice satisfies the scalar identity

$$\mathbf{C}_j^l[t]\mathbf{A}_j^l[t]\mathbf{B}_j^l[t-1] = \left(\mathbf{C}_j^l[t]\mathbf{B}_j^l[t]\right)\phi_j^l[t]. \tag{38}$$

The normalized one-trace recurrence is therefore

$$\widehat{\mathbf{E}}_j^l[t] = \phi_j^l[t]\widehat{\mathbf{E}}_j^l[t-1] + \mathbf{Q}_j^l[t] \in \mathbb{R}^{r_l \times 1}, \qquad \widehat{\mathbf{E}}_j^l[0] = \mathbf{0}_{r_l \times 1}. \tag{39}$$

Combining Equations (36) and (38) gives

$$\mathbf{C}_j^l[t]\mathbf{P}_j^l[t] \approx \left(\mathbf{C}_j^l[t]\mathbf{B}_j^l[t]\right)(\widehat{\mathbf{E}}_j^l[t])^\top \in \mathbb{R}^{1 \times r_l}. \tag{40}$$

Equivalently, substituting the trace recurrence into the previous display gives the exact visible error identity

$$\mathbf{C}_j^l[t]\mathbf{P}_j^l[t] - \left(\mathbf{C}_j^l[t]\mathbf{B}_j^l[t]\right)(\widehat{\mathbf{E}}_j^l[t])^\top = \mathbf{C}_j^l[t]\mathbf{A}_j^l[t]\mathbf{R}_j^l[t-1] \in \mathbb{R}^{1 \times r_l}, \qquad t \geq 2. \tag{41}$$

This identity gives the geometric meaning of $\widehat{\mathbf{P}} = \mathbf{B}\widehat{\mathbf{E}}^\top$. Each column of $\mathbf{P}_j^l[t-1]$ is a state-sensitivity vector in $\mathbb{R}^{d_l \times 1}$, while the factorization restricts all columns to the one-dimensional state subspace spanned by $\mathbf{B}_j^l[t-1]$. The equality holds for $d_l = 1$; for $d_l > 1$ the factorization is a rank-one state-axis compression. The compression is accurate when the transported residual has a small visible projection, $\|\mathbf{CAR}\|$, relative to the retained visible sensitivity. It is exact for a one-state neuron and at the first injection from a zero initial sensitivity. For multiple state variables, this is a rank-one state-axis compression whose accuracy is set by the visible projection of the transported

residual; visible matching leaves $\mathbf{R}$ as a separate error term. If $\eta_j^l[t] \in \mathbb{R}$ denotes the derivative of the current objective with respect to $Y_j^l[t]$, the corresponding row gradient estimate has shape $1 \times r_l$:

$$\widehat{\mathbf{g}}_j^l[t] = \eta_j^l[t] \left(\mathbf{C}_j^l[t]\mathbf{B}_j^l[t]\right) \left(\widehat{\mathbf{E}}_j^l[t]\right)^\top. \tag{42}$$

For the additive LIF and CLIF currents used here, $\mathbf{CB} = 1$, so Equation (42) reduces to the usual learning-signal times eligibility product; any effective current scale is already contained in $\mathbf{Q}$ and hence in $\widehat{\mathbf{E}}$. The layer operator $\mathcal{P}_l$ sums rows for a dense projection and performs the corresponding cross-correlation for a convolution.

The bias sensitivity has the same state-side factor with a scalar trace:

$$\mathbf{P}_{j,b}^l[t] = \mathbf{A}_j^l[t]\mathbf{P}_{j,b}^l[t-1] + \gamma_j^l[t]\mathbf{B}_j^l[t], \qquad \mathbf{P}_{j,b}^l[t] \in \mathbb{R}^{d_l \times 1}, \quad \mathbf{P}_{j,b}^l[0] = \mathbf{0}_{d_l \times 1}, \tag{43}$$

$$F_j^l[t] = \phi_j^l[t]F_j^l[t-1] + \gamma_j^l[t], \qquad F_j^l[t] \in \mathbb{R}, \quad F_j^l[0] = 0. \tag{44}$$

Under the same factorization, $\mathbf{P}_{j,b}^l[t] \approx \mathbf{B}_j^l[t]F_j^l[t]$. For an identity current map, $\gamma_j^l[t] = 1$.

If the projection current has $u_l > 1$ channels or the visible state has $m_l > 1$ coordinates, the visible tangents $\mathbf{D}_j^l[t] = \mathbf{C}_j^l[t]\mathbf{A}_j^l[t]\mathbf{B}_j^l[t-1]$ and $\mathbf{H}_j^l[t] = \mathbf{C}_j^l[t]\mathbf{B}_j^l[t]$ become $m_l \times u_l$ matrices, a single scalar generally cannot match them exactly, and the least-squares coefficient replaces Equation (37) by the ratio $\langle \mathbf{H}_j^l[t], \mathbf{D}_j^l[t]\rangle_F / \|\mathbf{H}_j^l[t]\|_F^2$, where $\langle \mathbf{U}, \mathbf{V}\rangle_F \triangleq \operatorname{tr}(\mathbf{U}^\top\mathbf{V})$. These multi-channel extensions are outside the one-trace rule evaluated in this paper.

Visible matching first gives a coefficient $\phi_j^l[t] \in \mathbb{R}$ for each unit. To keep one input-shaped trace inside the projection Jacobian, its recurrence must use a common coefficient across postsynaptic units, as in Appendix A.3. Equal-weight fitting of the per-unit coefficients selects

$$\phi^l[t] = \frac{1}{n_l}\sum_{j=1}^{n_l} \phi_j^l[t] \in \mathbb{R}. \tag{45}$$

The production trace $\widehat{\mathbf{E}}^l[t]$ has shape $r_l \times 1$ for a dense layer and the same tensor shape as the actual projection input $\mathbf{A}^{l-1}[t]$ for a convolution. Sharing this trace across j is a second approximation, distinct from the state-direction factorization in Equation (33).

Fitting unnormalized visible tangents instead gives a weighted coefficient. With $\mathbf{H}_j^l[t]$ and $\mathbf{D}_j^l[t]$ of shape $m_l \times u_l$ as defined above, the joint least-squares fit is

$$\phi_{\text{vis}}^l[t] = \arg\min_{\varphi \in \mathbb{R}} \sum_j \|\mathbf{D}_j^l[t] - \varphi\mathbf{H}_j^l[t]\|_F^2 = \frac{\sum_j \langle \mathbf{H}_j^l[t], \mathbf{D}_j^l[t]\rangle_F}{\sum_j \|\mathbf{H}_j^l[t]\|_F^2}. \tag{46}$$

The denominator must be positive. For scalar currents and visible states, this weights each $\phi_j^l[t]$ by $(\mathbf{C}_j^l[t]\mathbf{B}_j^l[t])^2$. For the additive LIF and CLIF models, $\mathbf{C}_j^l[t]\mathbf{B}_j^l[t] = 1$, so the joint visible fit equals the arithmetic mean in Equation (45). For general models, coefficient fitting and visible-tangent fitting define different objectives.

For a one-state LIF neuron, $d_l = 1$, $\mathbf{C}_j^l[t] = \mathbf{B}_j^l[t] = 1$, and $\mathbf{A}_j^l[t] = \Phi_j^l[t]$. Equations (37) and (39) first reduce to the exact per-unit LIF recurrence. Applying the layer mean in Equation (45) and sharing the resulting trace across j gives the production DPET recurrence, with the second approximation described above. All Jacobians in this state-space form use the same SG substitution as the main text.

A.4.1 CLIF Example

The complementary LIF (CLIF) neuron provides a two-state example (Huang et al., 2024). For one unit, the state, current, visible state, and selector have shapes

$$\mathbf{Z}[t] = \begin{bmatrix} V[t] \\ M[t] \end{bmatrix} \in \mathbb{R}^{2\times 1}, \qquad X[t] \in \mathbb{R}, \qquad Y[t] = V[t] \in \mathbb{R}, \qquad \mathbf{C} = [1 \quad 0] \in \mathbb{R}^{1\times 2}. \tag{47}$$

The CLIF time constant is $\tau_c = 1/(1-\lambda) \in \mathbb{R}$. Its scalar dynamics are

$$S[t-1] = \Theta(V[t-1] - \vartheta) \in \{0,1\}, \qquad R[t-1] = \text{sigmoid}(M[t-1]) \in \mathbb{R}, \tag{48}$$

$$\overline{V}[t-1] = V[t-1] - S[t-1](\vartheta + R[t-1]) \in \mathbb{R}, \qquad V[t] = \lambda\overline{V}[t-1] + X[t] \in \mathbb{R}, \tag{49}$$

$$q[t] = \text{sigmoid}(V[t]/\tau_c) \in \mathbb{R}, \qquad S[t] = \Theta(V[t] - \vartheta) \in \{0,1\}, \tag{50}$$

$$M[t] = q[t]M[t-1] + S[t] \in \mathbb{R}. \tag{51}$$

The symbolic equations follow the main-text convention $\Theta(x) = 1$ for $x \geq 0$; the implementation uses a strict positive comparator, and the two conventions differ only at exact threshold equality. During differentiation, each spike derivative is replaced by the surrogate. Let $g[t] = \sigma'(V[t] - \vartheta) \in \mathbb{R}$, $r'[t-1] = R[t-1](1 - R[t-1]) \in \mathbb{R}$, and $q'[t] = q[t](1 - q[t])/\tau_c \in \mathbb{R}$.

The scalar $K[t]$ is the input tangent of the complementary state. With $K[0] = 0$ for the zero initial tangent, differentiating the last CLIF update with respect to the current input while holding the previous state fixed gives

$$K[t] \triangleq \frac{\partial M[t]}{\partial X[t]} = M[t-1]q'[t] + g[t] \in \mathbb{R}, \qquad \mathbf{B}[t] \triangleq \frac{\partial \mathbf{Z}[t]}{\partial X[t]} = \begin{bmatrix} 1 \\ K[t] \end{bmatrix} \in \mathbb{R}^{2\times 1}. \tag{52}$$

The state-transition Jacobian has shape 2×2:

$$\mathbf{A}[t] = \frac{\partial \mathbf{Z}[t]}{\partial \mathbf{Z}[t-1]} = \begin{bmatrix} A_{VV}[t] & A_{VM}[t] \\ K[t]A_{VV}[t] & q[t] + K[t]A_{VM}[t] \end{bmatrix} \in \mathbb{R}^{2\times 2}, \tag{53}$$

where

$$A_{VV}[t] = \lambda\,[1 - g[t-1](\vartheta + R[t-1])] \in \mathbb{R}, \qquad A_{VM}[t] = -\lambda S[t-1]r'[t-1] \in \mathbb{R}. \tag{54}$$

Since $\mathbf{CB}[t] = 1$, Equation (37) becomes

$$\phi^{\mathrm{CLIF}}[t] = \mathbf{CA}[t]\mathbf{B}[t-1] = A_{VV}[t] + A_{VM}[t]K[t-1] \in \mathbb{R} \tag{55}$$

$$= \lambda\,[1 - g[t-1](\vartheta + \mathrm{sigmoid}(M[t-1])) - S[t-1]r'[t-1]K[t-1]]\,. \tag{56}$$

The first term carries the membrane tangent. The second term carries the path through the complementary state. The implementation caches detached $K[t] \in \mathbb{R}$ for the next step. In a batch, $K_i^l[t]$ has one value per CLIF unit, so for a dense layer $\mathbf{K}_i^l[t] \in \mathbb{R}^{n_l \times 1}$ and for a convolution it has the same feature-map shape as $\mathbf{M}_i^l[t]$. $K[t]$ is a local state tangent. The eligibility representation contains one input-indexed trace $\widehat{\mathbf{E}}[t] \in \mathbb{R}^{r_l \times 1}$ per sample and projection. A deterministic double-precision gradient diagnostic verifies this extension against surrogate BPTT on a small linear projection; task results remain outside the main claim.

A.5 Batched, Convolutional, and Bias Gradients

For convolutional projections, $g'^l_{\mathbf{W}}$ remains linear in the input. The shared trace retains the input feature-map shape, the scalar decay averages output channels and spatial positions within each sample, and $\mathcal{P}_l$ becomes the convolutional weight-gradient cross-correlation. The following equations specify the shapes, batch reduction, and bias treatment.

Restoring the biases omitted from the main-text forward equations gives

$$\mathbf{X}^l[t] = g^l(\mathbf{S}^{l-1}[t]; \mathbf{W}^l) + \mathbf{b}^l, \qquad \mathbf{O}[t] = \mathbf{W}^o\mathbf{S}^L[t] + \mathbf{b}^o. \tag{57}$$

For a dense layer, $\mathbf{b}^l \in \mathbb{R}^{n_l}$; convolutional biases have one entry per output channel and broadcast over spatial positions. The readout bias has shape $\mathbf{b}^o \in \mathbb{R}^C$ and uses the ordinary current-step gradient. For a dense layer with the basic additive LIF current, define the scalar exact bias eligibility $F^l_{j,\mathrm{exact}}[t]$ by

$$F^l_{j,\mathrm{exact}}[t] \triangleq \frac{\mathrm{d}V_j^l[t]}{\mathrm{d}b_j^l} = \Phi_j^l[t]F^l_{j,\mathrm{exact}}[t-1] + 1 = \sum_{\tau=1}^{t}\prod_{s=\tau+1}^{t}\Phi_j^l[s], \qquad F^l_{j,\mathrm{exact}}[0] = 0, \tag{58}$$

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}b_j^l} = \sum_{t=1}^{T}\zeta_j^l[t]F^l_{j,\mathrm{exact}}[t]. \tag{59}$$

Here $\zeta_j^l[t]$ is the cut downstream learning signal defined in the main text. For a convolutional channel bias, the gradient additionally sums these contributions over spatial positions sharing that bias. The preceding state-space derivation extends the bias trace to general neurons and effective current transformations.

For a minibatch of N samples, we restore the sample index i omitted in the main text. In a dense layer, $\mathbf{Z}_i^l[t] \in \mathbb{R}^{n_l \times d_l}$ stores one row per unit, $\mathbf{A}_i^{l-1}[t] \in \mathbb{R}^{r_l \times 1}$ is the projection input,

$\widehat{\mathbf{E}}_i^l[t] \in \mathbb{R}^{r_l \times 1}$ is the shared trace, and $\boldsymbol{\delta}_i^l[t] \in \mathbb{R}^{n_l \times 1}$ is the current spatial learning signal. The dense weight gradient has shape $\mathbb{R}^{n_l \times r_l}$. For a convolution with c_l output channels and spatial axes $H_{l,1}, \ldots, H_{l,D_l}$, each of the d_l state coordinates has shape $\mathbb{R}^{c_l \times H_{l,1} \times \cdots \times H_{l,D_l}}$, so the stacked state has shape $\mathbb{R}^{d_l \times c_l \times H_{l,1} \times \cdots \times H_{l,D_l}}$ and the spatial signal has the corresponding feature-map shape. The trace and $\mathbf{Q}_i^l[t]$ have the actual projection-input tensor shape, and $\mathcal{P}_l$ returns the convolution-kernel shape. The layer coefficient $\phi_i^l[t]$ is scalar. Each sample maintains its own temporal coefficient and shared eligibility trace:

$$\phi_i^l[t] = \frac{1}{n_l} \sum_{j=1}^{n_l} \phi_{i,j}^l[t], \qquad \widehat{\mathbf{E}}_i^l[t] = \phi_i^l[t] \widehat{\mathbf{E}}_i^l[t-1] + \mathbf{Q}_i^l[t], \qquad \widehat{\mathbf{E}}_i^l[0] = \mathbf{0}_{r_l \times 1}. \tag{60}$$

For LIF units, $\phi_{i,j}^l[t] = \Phi_{i,j}^l[t]$ from Equation (3); for a multi-state unit, it is the visible matching coefficient in Equation (37). In the dense case, $\mathbf{Q}_i^l[t] \in \mathbb{R}^{r_l \times 1}$ is the current-to-weight gradient column: it equals the actual projection input $\mathbf{A}_i^{l-1}[t]$ for an identity current map and equals $\kappa_i^l[t]\mathbf{A}_i^{l-1}[t]$ when a detached scalar current scale is applied. In the convolution case, $\mathbf{Q}_{i,j}^l[t] \in \mathbb{R}^{r_l \times 1}$ denotes the local column for unit j, while the stored $\mathbf{Q}_i^l[t]$ has the actual projection-input tensor shape and supplies these local columns through $\mathcal{P}_l$. The shared-trace equation assumes this current-to-weight map is common across postsynaptic units j, as in the identity map and the common scalar scale computed over the current batch and spatial elements by the implementation. A unit-dependent map would require unit-specific current-to-weight columns and traces. The actual projection input includes any preceding normalization or pooling. The average over j includes postsynaptic spatial positions for convolutional layers but never the batch axis. For the batch-mean objective $\mathcal{J}_{\text{batch}}[t] = N^{-1} \sum_i \mathcal{J}_i[t]$, define $\boldsymbol{\delta}_i^l[t] = \partial \mathcal{J}_{\text{batch}}[t] / \partial \mathbf{V}_i^l[t] \in \mathbb{R}^{n_l \times 1}$ for the dense LIF/CLIF case. Since $Y = V$ for these neurons, this is the visible-state learning signal in the state-space derivation. The weight update is

$$\widehat{\nabla_{\mathbf{W}^l} \mathcal{J}_{\text{batch}}}[t] = \begin{cases} \displaystyle\sum_{i=1}^{N} \boldsymbol{\delta}_i^l[t] (\widehat{\mathbf{E}}_i^l[t])^\top \in \mathbb{R}^{n_l \times r_l}, & \text{dense,} \\ \displaystyle\sum_{i=1}^{N} \mathcal{P}_l(\widehat{\mathbf{E}}_i^l[t], \boldsymbol{\delta}_i^l[t]), & \text{convolution.} \end{cases} \tag{61}$$

The spatial errors already include the $1/N$ reduction, so no further division is needed. Autograd includes cross-sample dependencies from training-time BN in these errors because the derivative uses the batch objective. A custom backward substitutes the stacked traces for the projection inputs while retaining these spatial errors. For d_l-state dense units, full local sensitivity storage is $\mathcal{O}(N n_l d_l r_l)$. The shared input-indexed trace uses $\mathcal{O}(N r_l)$ values, while evaluating the per-unit state tangents uses $\mathcal{O}(N n_l d_l)$ state-sized quantities. For CLIF, the cached $\mathbf{K}_i^l[t]$ adds one state-sized tangent buffer, $\mathcal{O}(N n_l)$ in the dense case, and remains independent of T.

For the identity current map, the batch-shared bias estimate uses one scalar trace with $\widetilde{\phi}^l[t] = N^{-1} \sum_i \phi_i^l[t] \in \mathbb{R}$:

$$F^l[t] = \widetilde{\phi}^l[t] F^l[t-1] + 1, \qquad F^l[0] = 0, \tag{62}$$

$$\widehat{\nabla_{\mathbf{b}^l} \mathcal{J}_{\text{batch}}}[t] = F^l[t] \sum_{i=1}^{N} \mathcal{R}_l(\boldsymbol{\delta}_i^l[t]). \tag{63}$$

Here $F^l[t] \in \mathbb{R}$, $\mathcal{R}_l(\boldsymbol{\delta}_i^l[t])$ has the bias shape, and $\widehat{\nabla_{\mathbf{b}^l} \mathcal{J}_{\text{batch}}}[t]$ has that same shape: $\mathbb{R}^{n_l \times 1}$ for a dense layer or one value per output channel for a convolution. The operator $\mathcal{R}_l$ sums every within-sample axis that shares a bias and retains the output-feature or channel axis. Sharing the bias trace introduces an additional approximation. Storing one bias trace per sample removes this approximation but makes its storage grow with batch size.

A.6 Layer-Wise Reduction Error

This subsection specializes to the one-state LIF case. Thus $\Phi_j^l[t]$ is the LIF state-transition derivative from Equation (3), while $\phi_i^l[t]$ is its layer-wise reduction used in the DPET trace.

Let $\mathbf{E}_j^l[t] = [E_{j1}^l[t], \ldots, E_{jr_l}^l[t]]^\top \in \mathbb{R}^{r_l \times 1}$ collect the input-indexed entries $E_{jk}^l[t]$ into a column vector. Unrolling the within-layer recurrence in Equation (6) gives

$$\mathbf{E}_j^l[t] = \sum_{\tau=1}^{t} \left(\prod_{s=\tau+1}^{t} \Phi_j^l[s] \right) \mathbf{A}^{l-1}[\tau]. \tag{64}$$

Each historical activity is weighted by the intervening membrane-potential transitions. The empty product is one. For a dense layer, $\delta_j^l[t] \in \mathbb{R}$ and the j-th row of the layer-local weight estimate is $\delta_j^l[t](\mathbf{E}_j^l[t])^\top \in \mathbb{R}^{1 \times r_l}$. This retains only within-layer temporal paths.

The reduced DPET trace has the corresponding expansion

$$\widehat{\mathbf{E}}_i^l[t] = \sum_{\tau=1}^{t} \left(\prod_{s=\tau+1}^{t} \phi_i^l[s] \right) \mathbf{A}_i^{l-1}[\tau]. \tag{65}$$

For a dense layer, $\widehat{\mathbf{E}}_i^l[t] \in \mathbb{R}^{r_l \times 1}$ in this column-vector form. For a convolution, the same symbol denotes the trace tensor aligned with the projection input.

The layer mean in Equation (9) is the least-squares scalar approximation to the neuron-wise temporal derivatives:

$$\phi_i^l[t] = \arg\min_{\varphi} \sum_{j=1}^{n_l} \left(\Phi_{i,j}^l[t] - \varphi \right)^2. \tag{66}$$

This criterion minimizes one-step coefficient error. Weight-gradient error receives separate spatial weighting. For one sample in a dense layer with $\mathbf{W}^l \in \mathbb{R}^{n_l \times r_l}$, the difference between the gradient formed by neuron-wise eligibilities and the DPET trace is

$$[\Delta \mathbf{G}^l[t]]_{jk} = \delta_j^l[t] \left(E_{jk}^l[t] - \widehat{E}_k^l[t] \right). \tag{67}$$

Here $\Delta \mathbf{G}^l[t] \in \mathbb{R}^{n_l \times r_l}$. The spatial error weights neurons unequally, while Equation (66) treats them equally. The temporal products in Equations (64) and (65) can also accumulate the reduction error.

Table A1: Online methods differ in historical weight influence and later-loss feedback. Lagged distillation adds a causal auxiliary signal from completed minibatches; it does not recover the current sequence's future Jacobians.

Method	Historical weight influence	Later-loss feedback	Temporal graph
BPTT	exact postsynaptic dynamics	exact	full
SLTT (Meng et al., 2023)	current input only ($\phi = 0$)	omitted	none
OTTT$_\mathrm{A}$ (Xiao et al., 2022)	fixed decay ($\phi = \lambda$)	omitted	none
NDOT$_\mathrm{A}$ (Jiang et al., 2024)	layer-$(l-1)$ dynamics	omitted	none
DPET	reset-aware layer-l mean	omitted	none
+ lagged distillation	reset-aware layer-l mean	lagged auxiliary signal	none

A.7 NDOT Layer-Index Counterexample

NDOT's Equation (7) propagates the layer-l weight sensitivity with layer-l temporal factors, but its Equations (9) and (12) substitute factors from layer $l-1$ (Jiang et al., 2024). Consider a scalar postsynaptic LIF with fixed inputs $A[1] = 1$, $A[2] = 0$, weight w, zero initial state, and threshold one. Its states satisfy $V[1] = w$ and $V[2] = \lambda(V[1] - \Theta(V[1]-1))$. The surrogate weight eligibility at the second step is

$$E[2] = \lambda \left[1 - \sigma'(w-1) \right]. \tag{68}$$

For fixed w with $\sigma'(w-1) \neq 1$, changing the postsynaptic λ changes this eligibility while leaving the presynaptic history and its dynamics unchanged. Alternatively, at fixed λ, changing a subthreshold w changes the sigmoid surrogate factor without changing either input. A decay derived only from the presynaptic dynamics cannot reproduce both sensitivities in general. This shows that the proposed input filter does not generally reproduce the stated derivative. Its empirical performance remains a separate question.

B SUPPORTING COMPONENTS

B.1 LAGGED DISTILLATION SUPPLIES AUXILIARY FEEDBACK

The missing downstream term measures how current activity affects later predictions through deeper neuronal states. An online update cannot read later outputs from the sequence it is training on, so we instead read a teacher built from previously completed minibatches, extending temporal self-distillation across simulation steps (Dong et al., 2024). This is a classification-specific supporting method; the DPET trace itself is formulated without class labels or a task-specific loss.

Before each minibatch, we snapshot the teacher state $\mathcal{B}^- = \{\mathbf{b}^-, \mathbf{m}^{(2)-}, \mathbf{n}^-\}$. A cell indexed by class y and step $t < T$ stores a probability vector initialized to the uniform distribution, its second moment, and an effective sample count. The label distribution $\boldsymbol{\pi}_i$ lies on the probability simplex over the C classes: a one-hot vector for a hard label, and a convex combination of one-hot vectors under label smoothing or Mixup/CutMix. The historical teacher and its reliability are

$$\bar{r}_{i,t} = \sum_y \pi_{i,y} r_{y,t}, \qquad \mathbf{q}_{i,t} = \sum_y \pi_{i,y} \mathbf{b}^-_{y,t}, \tag{69}$$

$$\mathbf{p}_{i,t} = (1 - \bar{r}_{i,t})\boldsymbol{\pi}_i + \bar{r}_{i,t}\mathbf{q}_{i,t}, \quad \mathcal{L}_{\mathrm{KD}}[t] = \frac{\omega\xi^2}{N} \sum_i \bar{r}_{i,t}\, \mathrm{KL}\left(\mathbf{p}_{i,t} \,\|\, \mathrm{softmax}(\mathbf{O}_i[t]/\xi)\right). \tag{70}$$

The temperature is ξ, the loss weight is ω, and $r_{y,t} \in [0, 1]$ downweights cells with low count or high variance. An unseen cell has zero reliability. Counts and variance measure support and consistency, rather than teacher correctness. The terminal loss is $\mathcal{L}_{\mathrm{KD}}[T] = 0$.

After the sequence finishes, its detached later-step logits form the write target

$$\widetilde{\mathbf{p}}_{i,t} = \mathrm{softmax}\left(\frac{\sum_{s>t} \gamma^{s-t-1} \mathbf{O}_i[s]}{\xi \sum_{s>t} \gamma^{s-t-1}}\right), \tag{71}$$

$$\mathbf{b}^+_{y,t} = \eta \mathbf{b}^-_{y,t} + (1 - \eta) \frac{\sum_i \pi_{i,y} \widetilde{\mathbf{p}}_{i,t}}{\sum_i \pi_{i,y}}, \qquad t < T, \quad \sum_i \pi_{i,y} > 0. \tag{72}$$

The factor γ discounts later outputs and η controls the bank momentum. Only future minibatches can read the updated bank. There is no terminal-step bank cell.

Each class- and time-indexed cell stores teacher probability $\mathbf{b}_{y,t}$, second moment $\mathbf{m}^{(2)}_{y,t}$, and effective count $n_{y,t}$. Its reliability is

$$r_{y,t} = r_{\max} \frac{n_{y,t}}{n_{y,t} + n_0} \exp\left[-\kappa \sum_c \left(m^{(2)}_{y,t,c} - b^2_{y,t,c}\right)_+\right]. \tag{73}$$

The cap $r_{\max}$ bounds teacher influence, n_0 is the count prior, and κ controls the variance penalty. For a soft label $\boldsymbol{\pi}_i$, the implementation mixes class reliabilities using $\sum_y \pi_{i,y} r_{y,t}$. Let $M_{y,t} = \sum_i \pi_{i,y}$. For each cell with $M_{y,t} > 0$, the second moment and count update after the minibatch:

$$\mathbf{m}^{(2)+}_{y,t} = \eta \mathbf{m}^{(2)-}_{y,t} + (1 - \eta) \frac{\sum_i \pi_{i,y} \widetilde{\mathbf{p}}^2_{i,t}}{M_{y,t}}, \qquad n^+_{y,t} = n^-_{y,t} + M_{y,t}. \tag{74}$$

The criterion writes the bank after the final forward step and only for classes with positive label mass.

The resulting auxiliary gradient reaches the hidden layers through the current spatial graph:

$$\boldsymbol{\zeta}^l_{\mathrm{KD}}[t] = (\mathbf{D}^l_{t,t})^\top \nabla_{\mathbf{O}[t]} \big(w[t] \mathcal{L}_{\mathrm{KD}}[t]\big). \tag{75}$$

For the task and distillation losses, the current spatial signal becomes $\boldsymbol{\zeta}^l_{\mathrm{now}}[t] + \boldsymbol{\zeta}^l_{\mathrm{KD}}[t]$. The teacher carries information about historical later predictions, but it does not provide the current sample's future output Jacobians. In particular, $\boldsymbol{\zeta}^l_{\mathrm{KD}}[t]$ lies in the range of $(\mathbf{D}^l_{t,t})^\top$. A missing future component outside that range cannot be recovered by changing the current output target alone.

We therefore treat partial compensation of the missing downstream term as a mechanism hypothesis. A direct test holds the forward states and eligibility fixed and measures

$$\mathcal{E}_{\text{future}} = \left\| \boldsymbol{\zeta}^l_{\text{future}}[t] - \boldsymbol{\zeta}^l_{\text{KD}}[t] \right\|_2, \tag{76}$$

against $\|\boldsymbol{\zeta}^l_{\text{future}}[t]\|_2$ without distillation. A decrease would support partial compensation; accuracy alone would establish an auxiliary-objective benefit. The bank uses $\mathcal{O}(C^2T)$ storage for C classes, and cached detached logits use $\mathcal{O}(NCT)$. Neither object stores a temporal autograd graph.

B.2 Trace Placement and Normalization

Input linearity and BN placement. At a fixed time step, let $\mathbf{A}$ collect the batch's projection inputs and define the combined transform $h(\mathbf{A}; \mathbf{W}) = \mathrm{BN}(g(\mathbf{A}; \mathbf{W}))$. Its weight Jacobian satisfies

$$h'_{\mathbf{W}}(\mathbf{A}) = \mathbf{J}_{\mathrm{BN}}(g(\mathbf{A}; \mathbf{W}))\, g'_{\mathbf{W}}(\mathbf{A}),$$

where $\mathbf{J}_{\mathrm{BN}}$ is the input Jacobian of BN. Although $g'_{\mathbf{W}}$ is linear in $\mathbf{A}$, the training-time BN Jacobian depends on the projected activations, so $h'_{\mathbf{W}}$ is generally not linear in $\mathbf{A}$. The scalar factors in Equation (7) therefore cannot be absorbed into a single input-shaped trace, which rules out the input substitution in Equation (11), though not more expensive forward-sensitivity formulations. Moving BN after the neuron keeps it outside the projection-to-neuron mapping; alternatively, sWS preserves the required linearity because it reparameterizes weights independently of the activations (Brock et al., 2021). This restriction concerns the shared-input trace substitution, not eligibility-trace methods in general.

Normalization across time. Full-sequence SNN training commonly pools BN statistics across both batch and time, as in tdBN and TEBN (Zheng et al., 2021; Duan et al., 2022). For convolutional activations of shape $N \times T \times c \times H \times W$ with c channels, this computes each channel's statistics over $NTHW$ values. Our online setting instead normalizes the current $N \times c \times H \times W$ activations, over NHW values at each step, because pooling across all T steps would make current outputs depend on future activations. This distinction concerns the normalization scheme rather than BPTT itself: per-step statistics can also be used with BPTT (Kim & Panda, 2021). Per-step statistics remove cross-time dependence but do not resolve the trace incompatibility caused by pre-neuron BN.

Selected layers therefore use a DPET-compatible block, while the rest retain SLTT with conventional BN:

$$\begin{aligned} l \in \mathcal{I}_{\mathrm{DPET}}: &\quad \mathrm{Conv} \;\to\; \mathrm{LIF}_{\mathrm{DPET}} \;\to\; \mathrm{BN}, \\ l \notin \mathcal{I}_{\mathrm{DPET}}: &\quad \mathrm{Conv} \;\to\; \mathrm{BN} \;\to\; \mathrm{LIF}_{\mathrm{SLTT}}\,. \end{aligned} \tag{77}$$

The subscripts denote gradient rules; both blocks keep the forward LIF recurrence. A normalization-free projection with sWS is an alternative compatible configuration for selected layers (Brock et al., 2021), which we report separately from post-neuron BN. For each selected layer, the trace in Equation (8) accumulates the actual projection input $\mathbf{A}^{l-1}[t]$, including any preceding normalization or pooling, rather than assuming binary presynaptic spikes. Because relocating BN changes the forward computation, a selective configuration is a joint choice of gradient rule and normalization placement.

Layers differ in the temporal sensitivity that SLTT omits, in how well a shared trace approximates their eligibility, and in how much the normalization change costs them, so tracing every layer need not maximize accuracy. Unselected layers keep conventional BN and still propagate current spatial gradients to earlier layers; only $\mathcal{I}_{\mathrm{DPET}}$ allocates trace state, at

$$M_{\mathrm{trace}} = \mathcal{O}\left(N \sum_{l \in \mathcal{I}_{\mathrm{DPET}}} |\mathbf{A}^{l-1}| \right). \tag{78}$$

We freeze each architecture's mask before the reported evaluation and do not implement an adaptive selector. Separating the two effects requires two comparisons: the full selective configuration against conventional-BN SLTT measures the net deployment benefit, whereas changing only the gradient rule within a fixed forward configuration isolates the trace contribution. All-DPET and alternative fixed masks test whether the chosen allocation is preferable at comparable training budgets. Selective temporal structure itself is not new (Hu et al., 2024); what is specific here is the joint placement of strict-online traces and their compatible spatial blocks.

B.3 Membrane Regularization Provides Direct Local Feedback

A temporal trace is useful only through the learning signal multiplied into it. The task-gradient path through a spike contains its surrogate derivative. When that derivative is small, task supervision through this path can be weak even if historical eligibility is retained. We add a loss directly on the membrane potential to provide feedback to units outside a threshold-relative band. Membrane-potential regularization has prior uses, including reducing spike-quantization error (Guo et al., 2023). Our penalty instead targets distance from the firing threshold during a limited warm-up period.

For the regularized DPET layers $\mathcal{I}_{\mathrm{reg}} \subseteq \mathcal{I}_{\mathrm{DPET}}$,

$$R_{\mathrm{mem}}[t] = \frac{1}{|\mathcal{I}_{\mathrm{reg}}|} \sum_{l \in \mathcal{I}_{\mathrm{reg}}} \mathbb{E}_{i,j} \left[\mathrm{ReLU}\left(\frac{|V_{i,j}^{l}[t] - V_{\mathrm{th}}|}{|V_{\mathrm{th}}|} - m_v \right)^2 \right]. \tag{79}$$

The expectation averages samples and neuronal units within a layer. The margin m_v specifies the unpenalized band. For one unit, let $z = (V - V_{\mathrm{th}})/|V_{\mathrm{th}}|$ and $r(V) = (|z| - m_v)_+^2$. Away from the band boundary,

$$\frac{\partial r}{\partial V} = \frac{2}{|V_{\mathrm{th}}|}(|z| - m_v)_+ \operatorname{sign}(z). \tag{80}$$

This direct derivative reaches the regularized membrane without passing through that unit's spike surrogate. Gradients to earlier layers still follow their own spatial operators.

A detached running penalty controls the loss weight:

$$c_{e,u} = c_0 \left(1 - \frac{e}{E_w}\right)_+ \mathrm{clip}\left(\frac{\bar{R}_u}{R^\star}, s_{\min}, s_{\max}\right), \tag{81}$$

$$\bar{R}_u = \mu \bar{R}_{u-1} + (1-\mu)\,\mathrm{stopgrad}(R_{\mathrm{mem}}[t_u]), \qquad u > 1. \tag{82}$$

The parameters c_0, E_w, $R^\star$, μ, and $s_{\min}, s_{\max}$ set the base weight, warm-up duration, reference penalty, momentum, and scale bounds; e is the epoch and u counts regularizer evaluations at steps t_u, with $\bar{R}_1$ initialized from the first detached penalty. Larger band violations raise the bounded scale, and the epoch factor removes the penalty after warm-up, all without an extra model or temporal graph.

The penalty does not extend temporal memory, and the two effects oppose each other, since raising the surrogate derivative lowers the soft-reset factor Φ in Equation (3), which reaches zero at the threshold for $V_{\mathrm{th}} = 1$ and a sigmoid surrogate of slope 4. The regularizer therefore serves local optimization, while DPET tracks whatever state-dependent sensitivity results.

B.4 Optional K-Step Selective Backpropagation

Layer selection determines which layers use DPET; optional time-step selection determines which steps receive a backward pass. Following SLTT-K (Meng et al., 2023), we compute gradients at a preselected set $\mathcal{T}_K$ of K steps, while every step still advances all forward neuronal states and allocated traces. The stratified-cycle selector partitions $1{:}T$ into K strata and preselects one step per stratum. Setting $K = T$ recovers full-step training. The per-step objective combines the task, distillation, and membrane terms:

$$\mathcal{J}[t] = w[t]\,[\ell(\mathbf{O}[t], y) + \mathcal{L}_{\mathrm{KD}}[t]] + c_{e,u} R_{\mathrm{mem}}[t], \tag{83}$$

$$\widehat{\mathbf{g}}_K = \frac{T}{K} \sum_{t \in \mathcal{T}_K} \widehat{\nabla_\theta \mathcal{J}[t]}, \qquad |\mathcal{T}_K| = K. \tag{84}$$

Disabled losses contribute zero. Time indices are preselected so that regularizer collection and backward decisions never depend on future input from the current sequence. Combinations with membrane potential regularization use only preselected time steps; selectors that decide after the forward pass run without this regularizer. The T/K scaling reproduces the dense sum when per-step gradients agree, but the selected update need not equal the dense update for general temporal gradients. Carrying the trace through skipped steps preserves historical input sensitivity but does not

reconstruct the loss gradients those steps would have contributed. Each selected loss is backpropagated through its current-step graph, which is discarded before the next step; accumulated gradients are applied once per sequence. This extension reduces spatial backward calls without changing the trace-storage requirement.

An optional head-only warm-up restricts a subset $Q \subseteq \mathcal{T}_K$ of selected early steps to classifier updates. Before computing their logits, we detach the classifier input and use $\mathcal{J}_{\mathrm{head}}[t] = w[t][\ell(\mathbf{O}[t], y) + \mathcal{L}_{\mathrm{KD}}[t]]$. Let W denote the epoch window during which this warm-up applies. With feature-extractor parameters θ_b and classifier parameters θ_h, the schedule accumulates

$$\widehat{\mathbf{g}}_{\mathrm{head}} = \frac{T}{K}\left[\sum_{t\in\mathcal{T}_K\setminus Q} \nabla_{\theta_b,\theta_h}\mathcal{J}[t] + \sum_{t\in\mathcal{T}_K\cap Q} \nabla_{\theta_h}\mathcal{J}_{\mathrm{head}}[t]\right], \qquad e \in W. \tag{85}$$

Outside W, training returns to Equation (84). While membrane regularization is active, its penalty still updates the detached controller but contributes no backward gradient on these steps; all other selected steps use the complete current-step graph. This schedule tests whether restricting early backbone updates improves the accuracy–cost trade-off. Per-task schedules appear in Appendix C.3, and Appendix D.7 reports the measurements.

B.5 Complete Update Schedule

Algorithm 1 combines the two selection axes with the auxiliary signals. It holds parameters fixed for the sequence and applies one optimizer update at its end. Selected affine-current layers use Equation (11); unselected layers use ordinary current-step autograd, including their conventional BN. The selected bias trace $F^l[t]$ uses the batch-mean decay, with gradients reduced by $\mathcal{R}_l$, which sums axes sharing one bias. Appendix A.5 gives this additional bias approximation.

Only selected layers allocate presynaptic traces, and no autograd graph survives a time-step boundary. The teacher snapshot stays read-only throughout the minibatch. The complete state budget comprises neuronal states, selected eligibility traces, detached logits, the class–time teacher, and the scalar regularizer controller; trace storage is independent of T, whereas the teacher and logit buffers carry the T-dependence stated in Section 4.2. Full layer selection with $K = T$ and disabled auxiliary losses recovers dense DPET, while empty layer selection with conventional BN recovers SLTT.

C Experimental Protocol

C.1 Architectures and Neuron Dynamics

Table A2: Architectures used by the main experiments. VGG and MaxResNet main-path spatial convolutions use 3×3 kernels; MaxResNet downsampling uses 1×1, and Sequential CIFAR uses length-3 1D kernels.

Setting	Topology and readout	Neuron and normalization placement
CIFAR VGG11	Eight convolutions with channels $64, 128, 256, 256, 512, 512, 512, 512$; average pooling after blocks 2, 4, 6, and 8; dropout 0.1 and a linear classifier.	Eight projection→soft-reset LIF→BN blocks.
CIFAR10-DVS VGG11	The same channel sequence on $2 \times 48 \times 48$ frames; average pooling after blocks 2, 4, 6, and 8; dropout 0.1 and a linear classifier.	Eight projection→soft-reset LIF→BN blocks.
MaxResNet-18	A 64-channel stem followed by residual stages with $3, 3, 2$ blocks and channels $128, 256, 512$; global average pooling and a linear classifier.	Eight main projections use sWS→soft-reset LIF; the other activation neurons use the backbone's hard-reset LIF. Stem, second-convolution, and downsample paths retain BN.
Sequential CIFAR	Six 1D convolutional LIF blocks of width 768 with average pooling after blocks 3 and 6; one $6144 \to 1536$ LIF layer and a linear classifier. Each image column is one time step.	Projection→soft-reset LIF→BN without affine parameters; all traced layers use DPET.

The VGG and Sequential CIFAR DPET-equipped neurons (Table A2) use the soft-reset recurrence in Equation (1) with $\lambda = 0.5$, $V_{\mathrm{th}} = 1$, and a sigmoid SG with slope 4. Every online condition detaches its recurrent neuronal state between adjacent time steps. The VGG, DVS, and Sequential

Algorithm 1 Selective DPET with lagged feedback for one minibatch

Require: Fixed layer set $\mathcal{I}_{\mathrm{DPET}}$ and its normalization configuration; selected steps $\mathcal{T}_K$; head-only steps $Q \subseteq \mathcal{T}_K$; minibatch and parameters θ

Snapshot teacher state $\mathcal{B}^-$; retain the regularizer controller from earlier minibatches
Initialize all neuronal states, selected-layer traces $\widehat{\mathbf{E}}^l[0] = 0$, selected bias traces $F^l[0] = 0$, and $\widehat{\mathbf{g}} = 0$
for $t = 1, \dots, T$ **do**
  for $l = 1, \dots, L$ **do**
    if $l \in \mathcal{I}_{\mathrm{DPET}}$ **then**
      Compute $\phi_i^l[t]$ from previous membrane potentials using Equation (9)
    end if
    Run the configured forward block, preserving its LIF recurrence
    if $l \in \mathcal{I}_{\mathrm{DPET}}$ **then**
      Update $\widehat{\mathbf{E}}_i^l[t]$ and any $F^l[t]$ using Equations (8) and (63)
    end if
  end for
  If $t \in Q$, detach the classifier input before computing logits $\mathbf{O}[t]$
  Cache detached logits only when lagged distillation is enabled
  if $t \in \mathcal{T}_K$ **then**
    Read the KD target from $\mathcal{B}^-$; use zero KD if disabled or $t = T$
    While membrane regularization is active, evaluate its penalty and update the detached EMA controller
    Form $\mathcal{J}[t]$, or $\mathcal{J}_{\mathrm{head}}[t]$ without the membrane penalty if $t \in Q$
    Spatially backpropagate to obtain current parameter gradients $\mathbf{h}[t]$ and errors $\boldsymbol{\delta}^l[t]$
    if $t \notin Q$ **then**
      Replace selected projection weight and bias gradients in $\mathbf{h}[t]$ by Equations (11) and (63)
    end if
    $\widehat{\mathbf{g}} \leftarrow \widehat{\mathbf{g}} + (T/K)\mathbf{h}[t]$; head-only steps contribute classifier gradients only
  end if
  Detach neuronal states and traces; discard the current-step graph
end for
Update teacher cells for $t < T$ and positive class mass from detached logits, if enabled
Apply one optimizer update to θ using $\widehat{\mathbf{g}}$

CIFAR runs set `detach_reset=false`, while the MaxResNet runs set it to true for all neurons; neither setting reconstructs a temporal autograd path. The manual DPET recurrence retains the reset term in both cases. MaxResNet's remaining activation neurons use a hard reset with $\tau = 2$, input decay enabled, and detached state. Thus, the MaxResNet rows represent hybrid DPET/SLTT training in eight convolutional layers.

C.2 Frozen Selective Configurations

Selective placement uses the index of an eligible projection-neuron block, not an arbitrary layer count; each mask is fixed per architecture before evaluation and does not encode an optimal selection rule. MaxResNet-18 traces the eight eligible main projections, its full eligible set; TinyImageNet VGG-13 selects five of ten convolutional blocks, at zero-based indices $1, 3, 5, 7, 9$. Both use scaled weight standardization with a calibrated initialization and no intervening normalization, while their unselected blocks retain conventional BN and SLTT. CIFAR VGG11 and DVS trace all eight blocks with post-neuron BN; Spiking PointNet traces all projections without normalization; SDT traces its MLP owners and SpikingLM its Q/K/V projections, with SLTT elsewhere (Appendix D).

C.3 Training Configurations

The VGG, DVS, and Sequential CIFAR runs use the per-step TET objective (Deng et al., 2022)

$$\ell_{\mathrm{TET}}(\mathbf{O}[t], y) = (1 - \beta)\ell_{\mathrm{CE}}(\mathbf{O}[t], y) + \beta\,\mathrm{MSE}(\mathbf{O}[t], \mathbf{1}), \tag{86}$$

where $\mathbf{1}$ is the all-one logit target; MaxResNet uses cross-entropy without the TET term. Table A3 lists the optimization settings shared by the main accuracy matrix, and the paragraphs below give the data protocols and augmentation recipes.

Table A3: Training configurations for the main accuracy matrix. AGC denotes adaptive gradient clipping, and K is the optional backward-step budget of Appendix B.4.

	CIFAR-10/100 (VGG11)	CIFAR10-DVS (VGG11)	CIFAR-100 (MaxResNet-18)	Seq. CIFAR-100 (1D CNN)
Time steps T	6	10	4	32
Optional K	4	5	3	–
Epochs	300	150	410	300
Batch size	128	128	64	256
Precision	bf16	fp32	bf16	bf16
Optimizer	SGD (0.9)	SGD (0.9)	AdamW	SGD (0.9)
Peak learning rate	0.1	0.1	1.5×10^{-3}	0.1
Weight decay	0	5×10^{-4}	0.06	0
AGC	0.008	0.04	–	–
Schedule	cosine	cosine	warm-up+cosine	cosine
Per-step objective	TET, $\beta = 0.05$	TET, $\beta = 10^{-3}$	CE	TET, $\beta = 0.05$
Time weighting w_t	1	1	$1/T$	$1/T$

CIFAR-10/100. The VGG runs use direct encoding of CIFAR-10/100 (Krizhevsky, 2009) images. Training applies random crop and flip, CIFAR AutoAugment (Cubuk et al., 2019), Cutout-16 (DeVries & Taylor, 2017), and Mixup/CutMix (Zhang et al., 2018; Yun et al., 2019) with parameters $0.2/1.0$ and probability 1.

CIFAR10-DVS. Inputs are ten integrated CIFAR10-DVS (Li et al., 2017) frames under the original 9,000/1,000 split, and all 9,000 training examples participate in fitting. The official 1,000-example evaluation split is queried during training, and the test loader aliases that evaluation loader; results under this protocol are development values, not independent one-shot test results. Augmentation uses random resized crop, horizontal flip, CIFAR10DVSNDA (Li et al., 2022) with $(M, N) = (1, 2)$, and Mixup/CutMix with parameters $0.2/0.3$ and probability 0.5.

MaxResNet-18 on CIFAR-100. We train MaxResNet-18 with AdamW (Loshchilov & Hutter, 2019). Weight decay applies to non-bias parameters with more than one dimension, with 20 warm-up epochs and a 10-epoch cooldown around the cosine schedule. The data recipe uses horizontal flip with probability 0.5, RandAugment (Cubuk et al., 2020) `rand-m9-n1-mstd0.4-inc1`, constant-value random erasing with probability 0.25, Mixup 0.75, CutMix 0.5, and label smoothing 0.1; Mixup and CutMix stop after epoch 249.

The eight traced MaxResNet projections use sWS with no intervening normalization. We adapt the weight-scale component of SpikeInit (Shi & Yu, 2026) to the soft-reset LIF dynamics in Equation (1). For target firing rate $\rho = 0.1$ and horizon $T = 4$, the initializer samples 65,536 standard-Gaussian current sequences with seed 17071 and scales each current by $g\sqrt{\rho}$. A 48-step bisection selects the effective sWS scale g so that the simulated final-step firing rate matches ρ. The implementation initializes each learnable gain parameter to g/s with fixed sWS constant $s = 2.74$; the gain remains trainable.

Sequential CIFAR-100. A fixed 5,000-example subset of the CIFAR-100 training split supports checkpoint selection using seed 1729; the official test split remains unseen during training. Augmentation applies horizontal flip, TrivialAugmentWide (Müller & Hutter, 2021), random erasing with probability 0.1, and one Mixup/CutMix draw with parameters $0.2/1.0$ for every minibatch. The methods in Table 6 share projection→soft-reset LIF→post-neuron BN without affine parameters, $T = 32$, batch size 256, bf16, 300 epochs, and the same per-step TET objective. NDOT_A transports its presynaptic spike trace through the current BN scale and pooling; the current BN output contributes its offset once, while historical BN-statistic derivatives are omitted. Its bias trace uses the mean presynaptic decay coefficient. The first projection uses the current input as its trace, and the classifier uses ordinary current-step gradients. The pp-prop condition uses $\alpha = 0.9$ and the $1 - \alpha^t$ correction.

Membrane-potential initialization. DVS and Sequential CIFAR runs use MP-init: for training minibatch q, each enabled layer records the terminal pre-reset membrane mean and updates

$h^l_{0,q+1} = 0.9h^l_{0,q} + 0.1\,\mathbb{E}_{i,j}[V^l_{i,j}[T;q]]$ with $h^l_{0,1} = 0$; the next sequence sets $V^l[0] = h^l_{0,q+1}$ and derives its initial spike with Equation (1). Candidates are recorded only during training, synchronized across distributed workers, and stored with each checkpoint; evaluation contributes no candidate.

Auxiliary-objective hyperparameters. The VGG, DVS, Sequential CIFAR, and MaxResNet runs share the membrane penalty $m_v = 0.5$, $R^\star = 1$, $\mu = 0.95$, and $[s_{\min}, s_{\max}] = [0.25, 4]$, with $(c_0, E_w) = (0.03, 40)$ for VGG and MaxResNet and $(0.1, 20)$ for DVS. Lagged temporal distillation uses $(\eta, \gamma, \xi, \omega, r_{\max}, n_0, \kappa) = (0.9, 1, 2, 1, 1, 32, 1)$ across the image, point-cloud, and language runs.

Backward-step schedules. All main rows hold architecture, augmentation, optimizer, schedule, instantaneous-objective scale, and update timing fixed within a column. Optional K-step rows use stratified-cycle sampling and T/K objective rescaling with the (T, K) pairs in Table A3; Spiking PointNet uses $(4, 3)$. These rows train the first time step through the classifier head alone during epochs $[10, 20)$; the fixed-step rows do not.

Checkpoint selection. The CIFAR VGG and MaxResNet families select the best epoch after repeated evaluation on the benchmark evaluation split, and the retained DVS protocol follows the same repeated-evaluation convention. Sequential CIFAR uses the separate selection subset described above and evaluates each selected checkpoint once on the official test split.

C.4 LIF Coefficient Distribution Analysis

Figure 2 visualizes the neuron-wise temporal coefficients $[\mathbf{\Phi}^l[t]]_{jj}$ in selected layers of Spiking VGG11 on CIFAR-100. All LIF neurons use $\lambda = 0.5$ and $V_{\text{th}} = 1$, with the same surrogate-gradient setting as the corresponding training runs. For each model, we collect the coefficients over 128 CIFAR-100 images and time steps $t = 2, \ldots, 6$. The initialization distributions aggregate three independently initialized models, while the trained distributions aggregate six checkpoints, comprising three DPET-trained and three SLTT-trained runs. Each initialization or checkpoint contributes equally to the displayed density. The dashed lines and bottom markers report the corresponding scalar projection $\phi^l[t] = \text{tr}(\mathbf{\Phi}^l[t])/n_l$, averaged over the same images, time steps, and model instances.

C.5 Exact Local-Eligibility Gradient Diagnostic

The diagnostic uses three SLTT-trained Spiking VGG11 checkpoints, one per training seed. These models trained on all 50,000 CIFAR-100 fitting images and selected checkpoints by repeated evaluation on the official evaluation split. The diagnostic uses fitting images and makes no held-out accuracy claim. Sampling seed 1729 fixes the first 64 entries of a training-set permutation, partitioned into four batches of 16. Images use the recorded CIFAR-100 normalization without augmentation. The forward graph uses post-neuron BN in training mode, dropout probability zero, zero initial membrane states, and six direct-encoding steps. All LIF neurons use $\lambda = 0.5$, $V_{\text{th}} = 1$, and a sigmoid surrogate of slope 4.

Every reference and estimator uses the sequence sum of $0.95\ell_{\text{CE}} + 0.05\ell_{\text{MSE}}$, with an all-one MSE target. Cross-entropy averages over the minibatch, and MSE averages over minibatch and class entries. The loss value and logit gradient match the production TET implementation exactly. We measure convolutional weight gradients before clipping or optimizer transformations. Biases, normalization parameters, and classifier parameters remain in the forward graph but do not enter the reported weight vectors. Forward and backward calculations use float32 on CPU; norm, dot-product, and error calculations use float64.

The SLTT graph supplies $\boldsymbol{\delta}^l[t] = \partial\ell_t/\partial\mathbf{V}^l[t]$ at the integrated membrane potential before the current spike. We detach each recorded $\boldsymbol{\delta}^l[t]$ from its computation graph and record the input sequence of each projection. Detaching preserves its numerical value; it neither sets the error to zero nor makes different time steps share one value. The reference keeps neuron-specific eligibilities under the same surrogate derivative, instead of reducing temporal derivatives to a layer mean. For flattened weights

$\mathbf{w}^l = \mathrm{vec}(\mathbf{W}^l)$, it evaluates

$$\mathbf{g}^l_{\mathrm{loc}} = \sum_t \left(\frac{\partial \mathbf{V}^l[t]}{\partial \mathbf{w}^l} \right)^\top \boldsymbol{\delta}^l[t]. \tag{87}$$

The Jacobian holds the presynaptic input sequence fixed and retains the complete within-layer LIF recurrence; the membrane vector includes all minibatch entries and spatial positions, so shared convolutional weights receive their full within-layer contributions. For each layer separately, a local convolution–LIF replay holds these inputs and other parameters fixed, retains the complete local recurrence, and evaluates this vector–Jacobian product. The replay passes the recorded errors as the vector argument of the VJP, so differentiation acts only on the replayed membrane states and never on the error signals themselves. The errors remain those of the network's current-step task loss, shared across the compared eligibility calculations. The reference is an exact evaluation of the within-layer eligibility contribution under that signal, not the complete gradient of the original network loss. Two degenerate cases validate the reference: detaching the previous membrane state reproduces SLTT, and detaching only the reset-spike derivative reproduces $\mathrm{OTTT_A}$, both to numerical precision. At the final LIF layer, where no downstream temporal recurrence remains, it likewise matches full surrogate BPTT.

For each estimator X, local relative error is $\|\mathbf{g}_X - \mathbf{g}_{\mathrm{loc}}\|_2 / \|\mathbf{g}_{\mathrm{loc}}\|_2$. Full-BPTT relative error instead uses $\mathbf{g}_{\mathrm{BPTT}}$ as reference.

All eight layers and all methods remain in the analysis; no sample is excluded and no denominator is regularized. Each checkpoint averages four fixed minibatches first. Aggregate values give equal weight to the eight layers and report the mean and sample standard deviation across the three training seeds. The layer–minibatch–checkpoint comparisons are paired measurements, not independent training trials.

D Additional Benchmark Protocols and Results

Most comparisons use three seeds; ImageNet and TinyImageNet use one seed each, and a single-seed value has no standard deviation. Validation and test metrics are never pooled.

D.1 ImageNet: Scale and Trace Allocation

Our matched SLTT and DPET runs on ImageNet-1K (Deng et al., 2009; Russakovsky et al., 2015) share the official 100-epoch $T = 1$ SLTT checkpoint, global batch 255, and 30-epoch $T = 6$ fine-tuning schedule. They use a triangular surrogate, scaled weight standardization, SGD with learning rate 0.001, zero weight decay, and cosine decay. Images use random resized crops to 224×224 and horizontal flips. The 16-layer DPET mask selects every legal first-convolution-to-neuron projection in the residual blocks and retains SLTT elsewhere (Table A4).

Table A4: ImageNet-1K context: NF-ResNet-34, $T = 6$, one seed, best validation accuracy (%). Published rows use their source protocols. Local rows share the official $T = 1$ checkpoint, global batch 255, and a cosine-30 fine-tuning schedule; the best row continues that scheduler to a plateau.

Source	Rule	Traced blocks	Auxiliary	Accuracy
Published (Meng et al., 2023)	SLTT	N/A	N/A	66.19
Published (Xiao et al., 2022)	$\mathrm{OTTT_A}$	N/A	N/A	65.15
Local matched	SLTT	N/A	N/A	65.53
Local matched	DPET	16	N/A	65.53
Local best	DPET	16	KD (0.1)	65.81

The matched local runs reach 65.53% with both SLTT and DPET, and a sweep over surrogate slope, distillation weight, membrane-regularization coefficient, auxiliary losses, and schedule length leaves the local result within 65.37–65.81% (Meng et al., 2023). We report ImageNet as scale evidence for the rule at this network depth, and the published rows are quoted under their own source protocols.

D.2 pp-prop: Single-Seed Eligibility Comparison

We implement BrainTrace's pp-prop rule (Wang et al., 2026) in our PyTorch projection–neuron blocks, with the corrected EMA bias denominator $1 - \alpha^t$. These runs evaluate the eligibility rule

under our VGG11 recipe, not the BrainTrace compiler or its published benchmark configurations. CIFAR-100 uses seed 2034, $T = 6$, 300 epochs, batch 128, FP32, post-neuron BN, soft-reset LIF with $\lambda = 0.5$, and zero initial membrane potential. The objective is per-step TET with coefficient 0.05; gradients accumulate over the six steps before each optimizer update. The recipe uses learning rate 0.1, adaptive gradient clipping 0.008, MixUp 0.2, and CutMix 1.0. Table 1 fixes $\alpha = 0.9$ for all reported pp-prop runs instead of tuning it per dataset, because any per-dataset choice would have to be made on the same benchmark evaluations used for reporting. The pp-prop and DPET runs use different seeds and numerical precision, so the two are compared as unpaired runs. The fixed-$\alpha = 0.9$ runs complete 300, 300, and 150 epochs on CIFAR-10, CIFAR-100, and CIFAR10-DVS, respectively; the DVS run uses corrected, serialized, training-only MP-init.

D.3 TinyImageNet: Transfer to CLIF

Table 1 in the main text reports the complete LIF/CLIF configuration comparison.

The TinyImageNet (Le & Yang, 2015) matrix uses 200 classes, VGG13, batch 128, six simulation steps, 300 epochs, bf16, per-step TET, and the same TrivialAugmentWide-based augmentation recipe (Müller & Hutter, 2021) for all conditions. No adaptive gradient clipping, model-weight EMA, or auxiliary objective is enabled. The selective mask uses five alternating convolutional blocks (zero-based indices 1, 3, 5, 7, and 9). Selected layers use scaled weight standardization and training-data initialization without activation normalization; unselected layers retain conventional BN. The all-layer variant applies this compatible configuration to all ten blocks. BPTT and SLTT retain conventional BN throughout. The experiment compares complete training configurations across neuron models, including their normalization placement.

D.4 Point Clouds: Validation and Test Separation

For ScanObjectNN (Uy et al., 2019), we use PB_T50_RS with 1,024 points, four simulation steps, and a deterministic class-stratified 10% holdout from the official training split. The official test split is separate. We train for 200 epochs with batch 48, learning rate 0.001, weight decay 10^{-4}, uniform time weighting, and normalization-free neuronal blocks. The BPTT and SLTT controls use three validation seeds and the same training-holdout checkpoint selection as DPET. Table 2 reports official-test accuracy for the best training-holdout checkpoint of each Spiking PointNet run, evaluated on all 2,882 test samples. Test evaluations use batch 48, bf16 autocast, and a single mean reduction over the four time-step logits, with no augmentation or voting. The $\mathrm{OTTT_A}$, $\mathrm{NDOT_A}$, and fixed-α pp-prop rows in Table 2 use the same Spiking PointNet protocol and best-holdout checkpoint selection.

The SDT comparison adapts the released Spiking Discrepancy Transformer (Lu et al., 2026) code with learned position encoding, four simulation steps, batch 32, float32, and 300 epochs. Its released training protocol repeatedly evaluates the official test split and selects the highest-scoring epoch there, and all SDT conditions here follow that same protocol. The three-seed BPTT, SLTT, and DPET results in Table 2 use seeds 2025, 2026, and 7240. BPTT and SLTT keep the reference BN placement, while DPET uses compatible normalization-free MLP trace owners. The BPTT row was produced with the released source code. The SDT $\mathrm{OTTT_A}$, $\mathrm{NDOT_A}$, and pp-prop controls are likewise test-selected and use the DPET-compatible MLP placement. The membrane penalty uses initial coefficient 0.01 and a six-epoch warm-up, with margin 0.5, target penalty one, EMA momentum 0.95, and scale bounds $[0.25, 4]$.

D.5 Matched Training-Cost Profiles

Table 7 measures one complete sequence update on fixed, preloaded training minibatches. Within each workload, all rules use the same inputs, objective, optimizer, and update timing. The CIFAR-100 VGG profile uses an RTX 3090, batch 128, bf16, and $T \in \{4, 8, 16, 32\}$. Each condition runs in three independent processes with ten warm-up updates and 100 timed updates per process. The PyTorch DPET condition changes only the fused LIF dispatch in the frozen Triton DPET source; both paths match forward outputs, and their bf16 gradients satisfy the recorded rounding-tolerance check.

Table A5: Relative L_2 gradient error on three DPET+LTD-trained VGG11 checkpoints (CIFAR-100, $T = 6$). Each seed averages four minibatches and eight layers; entries are mean±std across seeds.

	TET reference		TET+LTD reference
Method	Exact local	Full BPTT	Full BPTT
DPET	$0.1560_{\pm 0.0128}$	$0.3400_{\pm 0.0158}$	$0.4042_{\pm 0.0164}$
DPET+LTD	$0.2909_{\pm 0.0211}$	$0.3497_{\pm 0.0178}$	$0.3392_{\pm 0.0153}$

The SDT profile uses an RTX 5090, batch 32, fp32, and the first 32 ScanObjectNN training point clouds. All five rules share the same learned-position-encoding backbone and projection–PLIF–BN order at the five MLP owners; this controlled topology differs from the retained SDT accuracy configurations. Each condition runs in three independent processes with three warm-up updates and ten timed updates per process. The SDT Triton backend fuses membrane integration, spike generation, reset, and surrogate-gradient backward computation for all 44 PLIF neurons, including gradients for their learnable time constants. The five selected MLP owners also use Triton eligibility-trace updates; projection, normalization, geometric operations, and AdamW (Loshchilov & Hutter, 2019) use the same library implementations as the PyTorch condition. All timed updates include model-internal FPS/kNN, forward computation, loss, backward computation, gradient clipping, and AdamW; they exclude data loading, point-cloud preprocessing, and host-to-device transfer. CUDA synchronization brackets each timed update. We report peak allocated bytes; the raw logs also contain allocator-reserved bytes. For $T > 4$, the profile repeats the same prepared point cloud to measure resource scaling outside the trained SDT horizon. Figure 3 uses BPTT, fused Triton DPET, and fused Triton DPET+LTD from the same RTX 5090 measurement run as the SDT rows of Table 7. LTD uses the lagged class-time teacher bank with weight one; three warm-up updates populate the bank before ten measured updates in each of three processes. The LTD-minus-DPET peak differences are 0.015, 0.031, 0.062, and 0.133 MiB at $T = 4, 8, 16, 32$; BPTT completes $T = 4$ and runs out of memory in all three processes at each longer horizon. Colored point labels in Figure 3 show median peak allocated GiB, with five decimal places for the online rules. The two workloads use different hardware and precision, so Table 7 supports within-workload comparisons.

D.6 Gradient Diagnostic with LTD

This measurement repeats the diagnostic of Appendix C.5 with the auxiliary objective enabled, using three DPET+LTD-trained VGG11 checkpoints in place of the SLTT-trained family; all other controls follow that protocol. Within each seed, the weights and the frozen historical teacher bank are held fixed, so the two estimators differ only in whether the current-step objective includes the LTD term. Table A5 reports both estimators against three references: the exact local eligibility and full BPTT of the task-only TET objective, and full BPTT of the TET+LTD objective that DPET+LTD actually optimizes. Because the checkpoint family differs, these values are not paired with Table 5.

Against the task-only references, enabling LTD increases the error: the exact-local error rises from $0.1560_{\pm 0.0128}$ to $0.2909_{\pm 0.0211}$, and the full-BPTT error rises from $0.3400_{\pm 0.0158}$ to $0.3497_{\pm 0.0178}$. Against full BPTT of the TET+LTD objective, the ordering reverses: DPET+LTD reaches $0.3392_{\pm 0.0153}$, whereas the task-only estimator reaches $0.4042_{\pm 0.0164}$. Each rule is therefore closest to the full-BPTT gradient of the objective it optimizes, so the increase under the task-only reference reflects the changed objective rather than a degraded eligibility estimate. This diagnostic compares gradients under a fixed objective; it does not test whether the LTD signal compensates the missing future feedback. Separate frozen probes address that question: on three CIFAR-100 VGG checkpoints at $T = 6$, with the weights and the historical teacher bank held fixed, LTD gives a pooled residual ratio of $0.9887_{\pm 0.0059}$ relative to no LTD, while the mean of individual ratios is $1.0141_{\pm 0.0277}$. A class-only teacher performs similarly, so these probes do not establish an additional benefit from time-conditioned targets. They use fitting images and cover a small set of checkpoints.

D.7 Backward-Schedule Measurements

The optional K-step schedules yield mean accuracies of 94.84%, 76.92%, and 81.08% on CIFAR-10, CIFAR-100, and MaxResNet-18/CIFAR-100, respectively, compared with 94.89%, 76.97%, and 81.21% for full-step DPET. On CIFAR10-DVS, accuracy decreases from 84.13% to 83.67%; the

DVS head-only cell remains unmeasured and is excluded from conclusions. On Spiking PointNet, mean ScanObjectNN official-test accuracy decreases from 68.08% to 67.55%. These are three-seed development results. The K-step schedules therefore trade a small amount of accuracy for fewer backward passes.

D.8 Language: Full Fine-Tuning with Selective Traces

All four GLUE tasks (Wang et al., 2018) use the same released SpikingLM revision, maximum token length 128, and batch 32. SST-2, MRPC, and RTE train for 20 epochs; QNLI trains for 10. We fine-tune all parameters; only the eligibility rule is restricted to query/key/value projections. Remaining neuronal blocks use SLTT. The model retains its hard-reset LIF dynamics, external LayerNorm, and detached reset-spike convention. Writing $H[t] = \lambda V[t-1] + (1-\lambda)X[t]$ and $V[t] = H[t](1 - \mathrm{stopgrad}(S[t]))$, its local temporal factor is $\lambda(1 - S[t-1])$, with $\lambda = 0.5$. Q/K/V traces average this spike-gated factor; they do not use the soft-reset formula from the main derivation. Its pretraining is offline, so the online claim concerns downstream fine-tuning only. The online axis consists of four neuronal simulation steps, not autoregressive token processing. We report validation accuracy for every task, including MRPC; MRPC accuracy must not be compared directly with published F1 scores. All language rows in Table 3 use three seeds. The language-specific spike-rate penalty applies to Q/K/V activity during a two-epoch warm-up, with initial coefficient 10^{-3} and an unpenalized firing-rate band $[0.08, 0.24]$ around target 0.16. It replaces the membrane-potential penalty in Table 3 because that penalty does not transfer to pretrained SpikingLM. **Unlike vision LIF neurons trained from scratch, whose potentials remain within reach of the margin m_v throughout training, the pretrained Q/K/V neurons carry over a signed membrane distribution centered roughly three thresholds away from V_{th}, permanently outside that margin.** $R_{\mathrm{mem}}[t]$ is then persistently large, so its EMA $\bar{R}_u$ in Equation (82) stays above $R^\star$ and the clip saturates at $s_{\max}$: the coefficient locks at $c_0 s_{\max}$ (measured $0.01 \to 0.04$) instead of decaying with training. The resulting membrane gradient then reaches 3.4–$3.8\times$ the task loss and 1.6–$2.0\times$ the task gradient magnitude at near-zero cosine similarity to it, so it overrides the task direction rather than refining it. The spike-rate penalty avoids this failure mode: it regularizes the firing rate, a quantity bounded in $[0, 1]$ regardless of where the pretrained membrane potentials sit, rather than distance from threshold, and it leaves those membrane values untouched with no added backward pass.

E Limitations and Future Work

Limitations. First, although we confirmed that LTD improves accuracy on every benchmark where it is enabled, and although we measured its effect on the parameter gradient directly, we have not explained which component of the update it modifies. Its signal is a class-indexed bank assembled from completed minibatches, and the frozen probes of Appendix D.6 further find that it does not measurably reduce the missing future-feedback term. We therefore present it as an auxiliary objective with task-dependent utility, and we do not claim that it recovers the second challenge of Section 4. Second, although we evaluated DPET on image, event-based, point-cloud, and language benchmarks and reported multiple seeds for most of them, every target in these studies is categorical and the teacher is indexed by class, so continuous-valued objectives remain untested. The DPET trace is formulated without class labels or a task-specific loss, but that generality is unverified here, and for LTD the class-indexed bank rules out regression outright. Third, although we added a sequential benchmark in which temporal credit dominates the objective, and although DPET closes most of the gap to BPTT there, that evidence rests on one dataset, one architecture, and one horizon, and we have not measured how the benefit scales with the strength of temporal dependence. Our coverage is therefore widest on tasks whose time dimension carries the least task information, and narrowest on the tasks that would exercise the eligibility trace most directly.

Future Work. Three directions follow from these limitations. The first is to make LTD interpretable: decompose its contribution to the parameter update by layer and by training stage, and report how the distillation gradient aligns with the current-step and future components of the error signal as an explicit function of teacher design, so that the auxiliary objective can be designed rather than merely tuned. The second is to move beyond categorical targets, by evaluating DPET on continuous-valued objectives and by formulating a lagged teacher that does not require class labels. The third is a systematic study of sequential settings across datasets, architectures, horizons, and au-

toregressive generation, measuring how the advantage of a dynamics-derived eligibility trace scales with temporal dependence instead of reporting a single sequential benchmark.

F SOCIAL IMPACT

Lowering the training memory of spiking neural networks lowers the hardware barrier to training them, which we expect to widen access to neuromorphic research beyond groups that can afford the memory-intensive BPTT pipeline on multi-GPU hardware. Because the method leaves inference-time computation unchanged, its environmental benefit is confined to training and depends on the hardware and energy mix used there; we do not claim an inference-energy advantage. We do not see any negative societal impacts from this work.