NLT-PROP: NONLOCAL TEMPORAL CREDIT ASSIGNMENT FOR ONLINE LEARNING IN SPIKING NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Spiking neural networks (SNNs), inspired by biological neural networks, communicate through discrete spikes and offer the potential for energy-efficient information processing. They are commonly trained using backpropagation through time (BPTT) with surrogate gradients. However, BPTT requires propagating gradients backward in time, making it biologically implausible and incompatible with strictly online learning. Real-time recurrent learning (RTRL) enables online gradient computation, but maintaining full eligibility traces in recurrent networks requires memory that grows cubically with network width and incurs substantial computational costs. Many existing online methods reduce these costs by simplifying temporal dependencies between neurons, limiting their ability to capture nonlocal temporal influences. We propose NLT-prop, an online learning method that efficiently approximates nonlocal temporal credit assignment through recurrent connections. Starting from the forward recursion for eligibility traces, NLT-prop decomposes the temporal Jacobian into diagonal and off-diagonal components, separating local neuronal dynamics from interactions between neurons. It organizes temporal paths by their number of nonlocal transitions and uses structured low-rank approximations to propagate the retained contributions forward in time. For a fixed retained order, the memory required for eligibility traces scales linearly with network width. This formulation explicitly incorporates nonlocal temporal influences without storing the full activity history or introducing additional eligibility-trace decay parameters. Experiments on SHD, N-MNIST, CIFAR10-DVS, DVS Gesture, and DMTS demonstrate that NLT-prop achieves performance comparable to BPTT while supporting fully online learning.

1 INTRODUCTION

Spiking neural networks (SNNs) are biologically inspired models that process information through discrete spikes and dynamic neuronal states (Maass, 1997; Eshraghian et al., 2023). Their sparse communication offers the potential to reduce computational activity and energy consumption (Pfeiffer & Pfeil, 2018; Roy et al., 2019; Davies et al., 2021). By combining neuronal dynamics with trainable connectivity, SNNs provide a framework for both studying neural computation and developing machine learning systems.

However, training SNNs efficiently remains a central challenge. A widely used approach is backpropagation through time (BPTT) (Werbos, 1990), combined with surrogate derivatives for the spike function (Neftci et al., 2019; Zenke & Vogels, 2021; Eshraghian et al., 2023). BPTT unrolls the network over time and propagates errors backward through the resulting computation graph. This backward-in-time learning procedure has no counterpart in the causal operation of biological neural circuits (Bellec et al., 2020). It also requires storing intermediate neural states, causing memory consumption to grow linearly with sequence length (Xiao et al., 2022; Meng et al., 2023; Jiang et al., 2024). In full-sequence BPTT, parameter updates must wait for the sequence and its backward pass to finish, preventing updates as new observations arrive.

Computing temporal gradients online in the forward direction requires carrying the effects of past network activity into the current update. Real-time recurrent learning (RTRL) achieves this by recur-

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.

sively propagating state-to-parameter sensitivities (Williams & Zipser, 1989). Subsequent methods differ in how they represent and update this temporal information (Marschall et al., 2020). UORO, KF-RTRL, and the Optimal Kronecker-Sum (OK) and Kronecker Triple Product (KTP) approximations use randomized factorizations of the global sensitivity tensor (Tallec & Ollivier, 2018; Mujika et al., 2018; Benzing et al., 2019). In SNNs, e-prop and online spatio-temporal learning (OSTL) construct eligibility traces from neuronal and synaptic dynamics (Zenke & Ganguli, 2018; Bellec et al., 2020; Bohnstingl et al., 2023), while online training through time (OTTT), NDOT, and pp-prop use compact representations based on presynaptic activity, neuronal dynamics, and separable presynaptic and postsynaptic factors, respectively (Xiao et al., 2022; Jiang et al., 2024; Wang et al., 2026). Forward propagation through time (FPTT) formulates online updates through a dynamically regularized objective (Kag & Saligrama, 2021; Yin et al., 2023). These methods generally maintain learning states recursively and combine them with current learning signals, enabling parameter updates without a backward sweep through the full trajectory.

For methods based on forward sensitivities and eligibility traces, the storage complexity of these states is central to scalable online learning. In dense recurrent networks, exact RTRL requires cubic sensitivity storage, while UORO, KF-RTRL, and OK require quadratic storage at fixed approximation rank (Benzing et al., 2019; Marschall et al., 2020). KTP achieves linear storage through randomized tensor factorization (Benzing et al., 2019). Synapse-specific SNN traces also scale with the number of connections, whereas OTTT, NDOT, and pp-prop achieve linear trace storage using presynaptic activity traces, neuronal dynamics-based temporal traces, and separable presynaptic and postsynaptic factors, respectively (Xiao et al., 2022; Jiang et al., 2024; Wang et al., 2026). However, linear-memory SNN learning rules such as OTTT and pp-prop explicitly or implicitly neglect the off-diagonal contributions of the temporal Jacobian (Xiao et al., 2022; Wang et al., 2026). The off-diagonal Jacobian terms carry temporal credit across neurons. Removing them eliminates paths through which past activity influences subsequent network states via recurrent interactions. Preserving these nonlocal contributions within a linear eligibility-memory budget will therefore provide a route to reducing the storage cost of temporal credit assignment while retaining cross-neuron gradient information.

We propose NLT-prop, an online learning method that preserves nonlocal temporal credit through a low-rank approximation organized by nonlocal transition order. We split the temporal state Jacobian into local and nonlocal components, and expand the propagated sensitivities according to the number of cross-neuron transitions. The local eligibility traces are factorized into presynaptic and postsynaptic components. For each retained nonlocal order, a pair of vectors represents the propagated influence and shares the presynaptic factor with the local traces. The nonlocal Jacobian is approximated by an outer product with its diagonal removed, and each order is updated through deterministic rank-one compression. This construction supports online parameter updates with eligibility storage that scales linearly with the input and hidden widths for fixed approximation order, independently of sequence length.

The main contributions of this work are summarized as follows:

- We propose NLT-prop, an online learning method for SNNs that retains nonlocal temporal credit with linear eligibility-trace storage.
- We derive an expansion of temporal sensitivities by the number of cross-neuron transitions and develop a deterministic low-rank scheme to propagate and compress the resulting nonlocal contributions.
- We evaluate NLT-prop on SHD, N-MNIST, CIFAR10-DVS, DVS Gesture, and a delayed match-to-sample (DMTS) task, obtaining competitive accuracy against BPTT and online learning baselines, Achieving state-of-the-art (SOTA) performance. Ablation experiments further demonstrate the benefit of retaining nonlocal temporal propagation.

2 Related Work

BPTT for SNNs. BPTT unfolds neuronal dynamics over successive time steps and propagates loss gradients backward through the resulting computation graph (Werbos, 1990). In SNNs, surrogate derivatives of the spike function enable gradient propagation through discrete firing events (Neftci et al., 2019; Zenke & Vogels, 2021; Eshraghian et al., 2023). Recent work has improved gradient es-

timation and reduced the cost of backward computation. Dspike adapts the shape and smoothness of surrogate derivatives during training to improve gradient estimates (Li et al., 2021), while temporal efficient training (TET) reweights gradient contributions through losses at individual time steps to improve optimization and generalization (Deng et al., 2022). To accelerate backward computation, Sparse Spiking Gradient Descent restricts gradient calculations to neurons whose membrane potentials lie near the firing threshold, reducing arithmetic operations and memory usage (Perez-Nieves & Goodman, 2021). Temporally truncated local BPTT combines shorter backward windows with local learning signals to reduce state storage and memory access (Guo et al., 2023). Rate-based backpropagation further approximates BPTT gradients using temporally averaged representations, compressing repeated backward computations into a single spatial backward pass per sequence (Yu et al., 2024). Spatial learning through time (SLTT) removes temporal gradient paths and performs spatial backpropagation at each time step, making its activation storage independent of sequence length (Meng et al., 2023). For neuromorphic computation, EventProp formulates temporal backpropagation through adjoint dynamics that transmit errors at spike times (Wunderlich & Pehle, 2021). Its implementation on SpiNNaker2 uses event packets to communicate spikes and error signals during on-chip training (Béna et al., 2025). However, standard full-sequence BPTT still retains a temporal computation graph and revisits past states in a separate backward pass. This incurs computation and memory costs that grow with sequence length and prevents parameter updates from being computed solely in the forward pass as new observations arrive.

Online Learning for SNNs. RTRL computes temporal gradients in the forward direction by recursively propagating sensitivities of the current network state to its parameters (Williams & Zipser, 1989). For a dense recurrent network with H hidden units, exact RTRL requires $O(H^3)$ sensitivity storage and $O(H^4)$ computation per time step (Benzing et al., 2019; Marschall et al., 2020). Approximations for recurrent neural networks (RNNs) reduce these costs through factorization or locality. UORO maintains a randomized rank-one approximation of the state-to-parameter sensitivity matrix (Tallec & Ollivier, 2018). KF-RTRL uses Kronecker factors to reduce approximation variance, and OK improves compression within this representation (Mujika et al., 2018; Benzing et al., 2019). Their sensitivity storage is quadratic in network width at fixed rank. KTP further reduces this storage to linear complexity using three vector factors while retaining multiplication by the full temporal Jacobian (Benzing et al., 2019). Random feedback local online (RFLO) learning instead drops nonlocal gradient terms and combines local eligibility traces with random error feedback (Murray, 2019).

For SNNs, SuperSpike, e-prop and OSTL derive eligibility traces from neuronal and synaptic dynamics and combine them with learning signals (Zenke & Ganguli, 2018; Bellec et al., 2020; Bohnstingl et al., 2023). Such synapse-specific traces scale with the number of connections. FPTT takes an objective-based approach, using dynamic regularization to support online optimization (Kag & Saligrama, 2021; Yin et al., 2023). Online training with postsynaptic estimates (OTPE) approximates the delayed influence of spikes on downstream neurons in feedforward SNNs, improving alignment with BPTT gradients (Summe et al., 2024). A complementary approach changes neuronal dynamics: the hybrid-driven leaky integrate-and-fire (HD-LIF) model modifies spike generation and reset to improve temporal gradient separability and alignment during online training (Hao et al., 2026). To obtain linear eligibility-trace storage, OTTT maintains presynaptic activity traces (Xiao et al., 2022), NDOT incorporates neuronal dynamics into temporal trace updates (Jiang et al., 2024), and pp-prop factorizes eligibility traces into presynaptic and postsynaptic components after a diagonal approximation of the temporal Jacobian (Wang et al., 2026). These compact SNN trace recursions carry temporal information locally within neurons and omit the off-diagonal contributions responsible for nonlocal temporal propagation. Consequently, they discard gradient paths through which past activity affects later states via interactions between neurons, restricting the credit available for learning dependencies carried by those paths. NLT-prop retains nonlocal contributions by organizing temporal sensitivities according to the number of cross-neuron transitions and compressing each retained order deterministically. Combined with separable local traces, this preserves nonlocal temporal credit with eligibility storage linear in the input and hidden widths at fixed approximation order.

3 PROPOSED METHOD

In this section, we introduce NLT-prop, an online learning method for SNNs with recurrent connections. NLT-prop retains nonlocal temporal credit through a compact representation of cross-neuron propagation, while keeping eligibility-trace storage linear in the input and hidden widths at a fixed approximation order.

The method separates local neuronal dynamics from nonlocal interactions in the temporal state Jacobian. Local eligibility traces are factorized into postsynaptic and presynaptic components, while nonlocal contributions are organized by the number of cross-neuron transitions and compressed recursively, as shown in Figure 1. At each time step, these factors are updated using the current neuronal state and then combined with the current learning signal to form parameter gradients. Temporal credit is therefore propagated forward, allowing parameter updates as observations arrive.

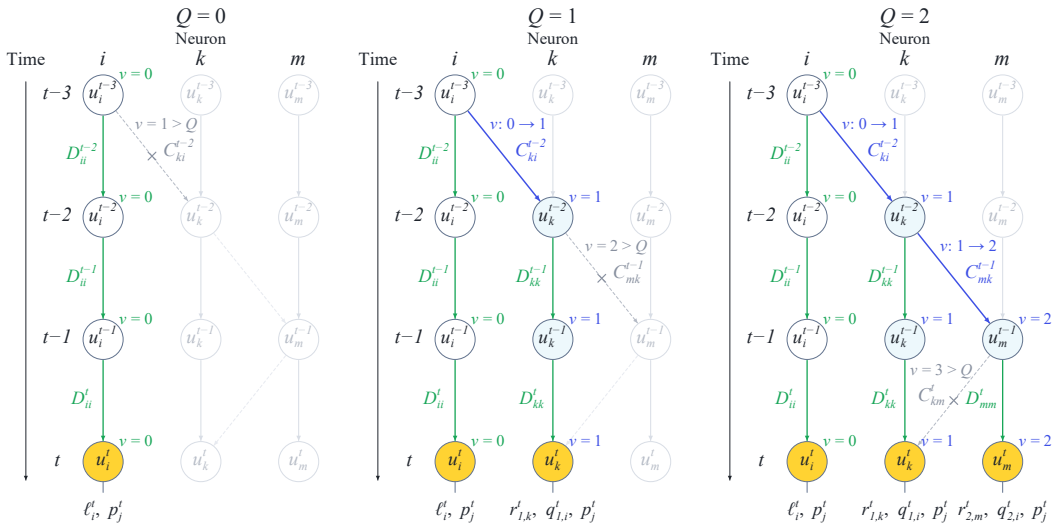


Figure 1: Illustrative temporal credit propagation for $Q = 0, 1, 2$. Same-neuron transitions D preserve the path order, while cross-neuron transitions C increase it by one. Contributions with order greater than Q are omitted.

3.1 DISCRETE LEAKY INTEGRATE-AND-FIRE (LIF) MODEL

Consider a simple layer of LIF neurons with lateral recurrent connections (Gerstner et al., 2014). Let $\boldsymbol{u}^t \in \mathbb{R}^N$ and $\boldsymbol{s}^t \in \{0,1\}^N$ denote the membrane potentials and spikes of N neurons at time t, and let $\boldsymbol{x}^t \in \mathbb{R}^M$ be the input. The recurrent and input weight matrices are $\boldsymbol{W} \in \mathbb{R}^{N\times N}$ and $\boldsymbol{V} \in \mathbb{R}^{N\times M}$, respectively. With leak factor $\lambda \in (0,1)$ and firing threshold $\vartheta > 0$, the dynamics with a subtractive reset are

$$\begin{aligned} \boldsymbol{u}^t &= \lambda \boldsymbol{u}^{t-1} + \boldsymbol{W}\boldsymbol{s}^{t-1} + \boldsymbol{V}\boldsymbol{x}^t - \vartheta \boldsymbol{s}^{t-1}, \\ \boldsymbol{s}^t &= H(\boldsymbol{u}^t - \vartheta \mathbf{1}), \end{aligned} \tag{1}$$

where H is the elementwise Heaviside function and $\boldsymbol{u}^0 = \boldsymbol{s}^0 = \mathbf{0}$. We use a triangular surrogate derivative for gradient computation (Neftci et al., 2019):

$$\psi_i^t = \gamma \max\left(0, 1 - \frac{|u_i^t - \vartheta|}{w}\right), \qquad \boldsymbol{\Psi}^t = \mathrm{Diag}(\boldsymbol{\psi}^t), \tag{2}$$

where $\gamma > 0$ and $w > 0$ control the height and width. All derivatives below use this surrogate, including the derivative through the reset.

3.2 Temporal State Jacobian

For a recurrent SNN governed by Eq. (1), the gradient separates into a current learning signal and a temporal sensitivity. Write $\boldsymbol{\Theta} = [\boldsymbol{W}\ \boldsymbol{V}]$, $\boldsymbol{\zeta}^t = [(\boldsymbol{s}^{t-1})^\top\ (\boldsymbol{x}^t)^\top]^\top$, and $g_k^t = \partial \mathcal{L}^t / \partial u_k^t$ for the loss $\mathcal{L}^t$ at time t. Define the eligibility matrix by $E^t_{k,(i,j)} = \partial u_k^t / \partial \Theta_{ij}$, with the parameter indices (i, j) flattened into columns. The gradient and forward recursion are

$$\begin{aligned} \frac{\partial \mathcal{L}^t}{\partial \Theta_{ij}} &= \sum_{k=1}^{N} g_k^t E^t_{k,(i,j)}, \\ \boldsymbol{E}^t &= \boldsymbol{J}^t \boldsymbol{E}^{t-1} + \boldsymbol{B}^t, \qquad B^t_{k,(i,j)} = \delta_{ki} \zeta_j^t. \end{aligned} \tag{3}$$

Here δ_{ki} is the Kronecker delta, and the temporal state Jacobian is $\boldsymbol{J}^t = \partial \boldsymbol{u}^t / \partial \boldsymbol{u}^{t-1} = \lambda \boldsymbol{I} + (\boldsymbol{W} - \vartheta \boldsymbol{I}) \boldsymbol{\Psi}^{t-1}$. This is the forward sensitivity recursion underlying RTRL (Williams & Zipser, 1989). NLT-prop replaces its $N \times N(N + M)$ eligibility matrix with a set of vector factors.

3.3 Diagonal Decomposition

We split the temporal Jacobian as $\boldsymbol{J}^t = \boldsymbol{D}^t + \boldsymbol{C}^t$, where $\boldsymbol{D}^t = \mathrm{Diag}(\boldsymbol{d}^t)$ with $d_i^t = \lambda + (W_{ii} - \vartheta)\psi_i^{t-1}$, and $\boldsymbol{C}^t = \mathrm{offdiag}(\boldsymbol{W})\boldsymbol{\Psi}^{t-1}$. The diagonal component propagates credit within each neuron, including its leak, reset, and self-connection. The off-diagonal component transfers credit between neurons.

With only diagonal propagation, $E^t_{0,k,(i,j)} = \delta_{ki} Z_{ij}^t$, where $\boldsymbol{Z}^t = \boldsymbol{D}^t \boldsymbol{Z}^{t-1} + \mathbf{1}(\boldsymbol{\zeta}^t)^\top$. We represent this local trace matrix by $\widehat{\boldsymbol{Z}}^t = \boldsymbol{\ell}^t (\boldsymbol{p}^t)^\top$, with a postsynaptic factor $\boldsymbol{\ell}^t \in \mathbb{R}^N$ and a shared presynaptic factor $\boldsymbol{p}^t \in \mathbb{R}^{N+M}$. Substituting the previous factorization gives a sum of two outer products. Its best rank-one approximation is computed through a 2×2 eigenproblem, following the Eckart–Young theorem (Eckart & Young, 1936). Denoting this projection by $\mathcal{P}_1$, the update and resulting local sensitivity are

$$\begin{aligned} \boldsymbol{\ell}^t (\boldsymbol{p}^t)^\top &= \mathcal{P}_1 \left[(\boldsymbol{d}^t \odot \boldsymbol{\ell}^{t-1})(\boldsymbol{p}^{t-1})^\top + \mathbf{1}(\boldsymbol{\zeta}^t)^\top \right], \\ \widehat{E}^t_{0,k,(i,j)} &= \delta_{ki} \ell_i^t p_j^t. \end{aligned} \tag{4}$$

The recurrent and feedforward inputs are compressed jointly, so both parameter groups share the same postsynaptic factor.

3.4 Off-Diagonal Decomposition

Expanding Eq. (3) by the number ν of off-diagonal transitions gives $\boldsymbol{E}^t = \sum_{\nu=0}^{t-1} \boldsymbol{E}_\nu^t$. The order-zero term follows the local recursion above, while $\boldsymbol{E}_\nu^t = \boldsymbol{D}^t \boldsymbol{E}_\nu^{t-1} + \boldsymbol{C}^t \boldsymbol{E}_{\nu-1}^{t-1}$ for $\nu \geq 1$. Each multiplication by $\boldsymbol{C}^t$ adds one cross-neuron transition. NLT-prop retains orders $1, \ldots, Q$ and represents their neuronal factors by $\widehat{\boldsymbol{H}}_\nu^t = \boldsymbol{r}_\nu^t (\boldsymbol{q}_\nu^t)^\top$.

We first estimate a dominant pair $\boldsymbol{a}^t, \boldsymbol{b}^t$ of $\boldsymbol{C}^t$ using a fixed number of deterministic power iterations, then set $\widetilde{\boldsymbol{C}}^t = \mathrm{offdiag}(\boldsymbol{a}^t (\boldsymbol{b}^t)^\top)$. The diagonal mask restricts this Jacobian approximation to cross-neuron transitions, and its products are evaluated directly from the two vectors. For the first nonlocal order, the source is a rank-one approximation $\boldsymbol{S}_1^t \approx \widetilde{\boldsymbol{C}}^t \mathrm{Diag}(\boldsymbol{\ell}^{t-1})$, obtained by power iteration. For higher orders, $\boldsymbol{S}_\nu^t = (\widetilde{\boldsymbol{C}}^t \boldsymbol{r}_{\nu-1}^{t-1})(\boldsymbol{q}_{\nu-1}^{t-1})^\top$. Each update again compresses two outer products. Sharing the current presynaptic factor across all orders yields

$$\begin{aligned} \widehat{\boldsymbol{H}}_\nu^t &= \mathcal{P}_1 \left[(\boldsymbol{d}^t \odot \boldsymbol{r}_\nu^{t-1})(\boldsymbol{q}_\nu^{t-1})^\top + \boldsymbol{S}_\nu^t \right], \qquad 1 \leq \nu \leq Q, \\ \widehat{E}^t_{k,(i,j)} &= \left(\delta_{ki} \ell_i^t + \sum_{\nu=1}^{Q} r^t_{\nu,k} q^t_{\nu,i} \right) p_j^t. \end{aligned} \tag{5}$$

Thus each retained order carries a distinct nonlocal contribution through two vectors, while a common presynaptic factor links these contributions to the synaptic parameters. The path expansion and compression steps are derived in Appendix A.

3.5 EXTENSION TO VGG-F

Contracting Eq. (5) with the learning signal gives the complete NLT-prop update. Splitting $\boldsymbol{p}^t = [(\boldsymbol{p}^t_{\mathrm{rec}})^\top\ (\boldsymbol{p}^t_{\mathrm{in}})^\top]^\top$ into recurrent and input factors, we obtain

$$\begin{aligned} \boldsymbol{z}^t &= \boldsymbol{g}^t \odot \boldsymbol{\ell}^t + \sum_{\nu=1}^{Q} \big((\boldsymbol{g}^t)^\top \boldsymbol{r}^t_\nu\big)\boldsymbol{q}^t_\nu, \\ \widehat{\nabla}_{\boldsymbol{W}}\mathcal{L}^t &= \boldsymbol{z}^t(\boldsymbol{p}^t_{\mathrm{rec}})^\top, \qquad \widehat{\nabla}_{\boldsymbol{V}}\mathcal{L}^t = \boldsymbol{z}^t(\boldsymbol{p}^t_{\mathrm{in}})^\top. \end{aligned} \tag{6}$$

The eligibility factors contain $(2Q+2)N + M$ scalars per sample. Their storage is $O(QN+M)$ and is independent of sequence length.

In VGG-F, feedback connects the final convolutional layer at time $t-1$ to the first layer at time t. This topology replaces an individual lateral transition with a traversal of the feedback loop. Let L be the number of layers, $\boldsymbol{A}^t_l = \partial \boldsymbol{u}^t_L / \partial \boldsymbol{u}^t_l$ the current spatial Jacobian, $\boldsymbol{F}$ the feedback convolution, $\boldsymbol{R}$ the resizing operator, and κ the spike scale. The loop operator is $\boldsymbol{G}^t = \boldsymbol{A}^t_1 \boldsymbol{F}\boldsymbol{R}\kappa\boldsymbol{\Psi}^{t-1}_L \approx \boldsymbol{a}^t(\boldsymbol{b}^t)^\top$, where $\boldsymbol{\Psi}^{t-1}_L$ contains the last layer's surrogate derivatives. Its diagonal is retained because a return to the same final-layer unit still traverses other layers. Accordingly, Q counts feedback-loop traversals.

Each convolutional layer stores a local pair $\boldsymbol{\ell}^t_l, \boldsymbol{p}^t_l$ and nonlocal pairs $\boldsymbol{r}^t_{l,\nu}, \boldsymbol{q}^t_{l,\nu}$, where $\boldsymbol{r}^t_{l,\nu}$ lies in the final-layer feature space and $\boldsymbol{q}^t_{l,\nu}$ in layer l. The local pair follows Eq. (4) with $\boldsymbol{d}^t_l = \lambda_l \mathbf{1} - \vartheta_l \boldsymbol{\psi}^{t-1}_l$ and the layer's presynaptic feature map. Define $\boldsymbol{b}^t_l = (\boldsymbol{A}^{t-1}_l)^\top \boldsymbol{b}^t$. The nonlocal updates become

$$\begin{aligned} \widehat{\boldsymbol{H}}^t_{l,\nu} &= \mathcal{P}^{(K)}_1\Big[(\boldsymbol{d}^t_L \odot \boldsymbol{r}^{t-1}_{l,\nu})(\boldsymbol{q}^{t-1}_{l,\nu})^\top + \boldsymbol{a}^t(\boldsymbol{v}^t_{l,\nu})^\top\Big], \\ \boldsymbol{v}^t_{l,\nu} &= \begin{cases} \boldsymbol{b}^t_l \odot \boldsymbol{\ell}^{t-1}_l, & \nu = 1, \\ \big((\boldsymbol{b}^t)^\top \boldsymbol{r}^{t-1}_{l,\nu-1}\big)\boldsymbol{q}^{t-1}_{l,\nu-1}, & \nu > 1. \end{cases} \end{aligned} \tag{7}$$

Here $\mathcal{P}^{(K)}_1$ denotes rank-one compression by K power iterations, also used for the VGG-F local pairs. The final-layer diagonal propagates the stored nonlocal modes, and the previous-step spatial Jacobian maps the loop's emission factor to each source layer. With current spatial learning signals $\boldsymbol{g}^t_l$, the convolutional gradient is

$$\boldsymbol{z}^t_l = \boldsymbol{g}^t_l \odot \boldsymbol{\ell}^t_l + \sum_{\nu=1}^{Q} \big((\boldsymbol{g}^t_L)^\top \boldsymbol{r}^t_{l,\nu}\big)\boldsymbol{q}^t_{l,\nu}, \qquad \widehat{\nabla}_{\boldsymbol{K}_l}\mathcal{L}^t = \mathrm{Corr}(\boldsymbol{p}^t_l, \boldsymbol{z}^t_l), \tag{8}$$

where Corr sums input–output products over positions sharing a kernel parameter. The feedback kernel uses the same construction with its input from the previous final-layer spikes. These factors retain linear eligibility storage in the feature-map sizes for fixed Q. Appendix A.8 gives the layer equations, operator products, and storage count.

4 EXPERIMENTAL DETAILS

4.1 DATASET DESCRIPTION

We evaluate NLT-prop on five datasets spanning auditory recognition, event-based vision, and working memory: SHD (Cramer et al., 2022), N-MNIST (Orchard et al., 2015), CIFAR10-DVS (Li et al., 2017), DVS Gesture (Amir et al., 2017), and DMTS based on the motion-direction protocol (Wang et al., 2026). SHD contains spike-encoded spoken digits, while N-MNIST, CIFAR10-DVS, and DVS Gesture involve handwritten digit, object, and gesture recognition from visual event streams, respectively. DMTS requires the network to determine whether two motion directions match across a variable delay containing distractors. For CIFAR10-DVS and DVS Gesture, we use VGG-F, which combines VGG-style convolutional blocks (Simonyan & Zisserman, 2015) with the feedback connection. SHD, N-MNIST, and DMTS use recurrent SNNs (RSNNs) with lateral connections. The compact spatial extent of N-MNIST allows its event frames to be flattened and processed by the same RSNN architecture family.

4.2 BASELINE

We compare NLT-prop with BPTT and four commonly used online learning methods for SNNs: OTTT (Xiao et al., 2022), OSTL (Bohnstingl et al., 2023), e-prop (Bellec et al., 2020), and pp-prop (Wang et al., 2026). These baselines cover temporal backpropagation, presynaptic activity traces, synapse-specific eligibility traces, and separable presynaptic and postsynaptic traces. OTTT and pp-prop have linear eligibility-trace storage, whereas the synapse-specific traces used by e-prop and OSTL scale with the number of connections, giving quadratic storage for dense recurrent layers. BPTT retains a temporal computation graph whose storage grows with sequence length. This comparison evaluates NLT-prop against both compact online trace representations and methods with larger temporal storage requirements.

4.3 EXPERIMENTAL SETTING

For each dataset, NLT-prop and all baselines use the same preprocessing, data splits, backbone architecture, and evaluation metrics. We retain the official splits for SHD, N-MNIST, and DVS Gesture. CIFAR10-DVS is randomly partitioned into training, validation, and test sets in an 8:1:1 ratio using a fixed seed. For DMTS, the three splits are generated independently using distinct random seeds. These partitions are fixed across methods and repeated runs. Baseline learning rules follow their original formulations and are adapted to the shared network topology and membrane-based readout. The resulting models have the same number of trainable parameters on each dataset. Full hyperparameter settings are provided in Appendix B.1.

4.4 COMPARISON OF PERFORMANCE

Table 1 summarizes the performance of NLT-prop and the five baselines. NLT-prop achieves the highest mean accuracy on SHD, CIFAR10-DVS, and DMTS, and ranks second on N-MNIST and DVS Gesture. On SHD and DMTS, it reaches accuracies of 76.83% and 78.62%, respectively. Compared with OTTT, these results represent gains of 6.63 and 11.54 percentage points, while the gains over pp-prop are 2.84 and 5.48 percentage points. The largest improvements over these two baselines with linear eligibility-trace storage occur on DMTS, where the network must retain the sample information across a variable delay containing distractors.

On CIFAR10-DVS, NLT-prop obtains 69.06% accuracy, exceeding OSTL at 68.94% and BPTT at 64.78%. On DVS Gesture, it achieves 93.41%, the highest accuracy among the online methods and 0.15 percentage points below BPTT. For N-MNIST, NLT-prop reaches 98.26%, compared with 98.49% for pp-prop and 95.70% for BPTT. Thus, NLT-prop exceeds BPTT in mean accuracy on four of the five datasets and remains close to it on DVS Gesture.

Across both RSNN and VGG-F backbones, these results show that NLT-prop combines competitive classification accuracy with online updates. Its trace representation supports auditory recognition, event-based visual recognition, and delayed-memory tasks.

4.5 ABLATION STUDY OF Q

We vary Q from 0 to 9 on SHD with the RSNN backbone and on DVS Gesture with VGG-F, using five seeds per configuration and keeping the remaining settings fixed within each sweep. Setting $Q = 0$ removes the nonlocal contributions and retains the local eligibility factors. On SHD, adding the first nonlocal order increases accuracy from $75.97 \pm 0.97\%$ to $76.83 \pm 1.08\%$, a gain of 0.86 percentage points. This is the highest mean accuracy across the tested orders. Higher orders give nonmonotonic results: accuracy falls to $72.42 \pm 6.19\%$ at $Q = 4$ and $71.37 \pm 5.51\%$ at $Q = 8$, while recovering to $76.51 \pm 0.83\%$ at $Q = 9$.

VGG-F also benefits from retaining nonlocal propagation. On DVS Gesture, accuracy increases from $92.27 \pm 0.63\%$ at $Q = 0$ to $93.33 \pm 1.91\%$ at $Q = 1$. The highest mean, $93.48 \pm 0.86\%$, occurs at $Q = 7$, giving a gain of 1.21 percentage points over $Q = 0$ and 0.15 percentage points over $Q = 1$. Increasing the order to $Q = 8$ reduces accuracy to $91.67 \pm 1.10\%$. Thus, the best order differs between the two backbones, while the first nonlocal order captures most of the observed improvement over local propagation in both.

Table 1: Performance comparison on five datasets. ACC is reported as mean ± sample standard deviation in percent over five runs. Params denotes trainable parameters. sWS denotes scaled weight standardization. For each dataset, the highest mean accuracy is shown in **bold**, and the second-highest is underlined.

Dataset	Method	Network structure	Params	ACC (%, Mean±SD)
SHD	BPTT	RSNN	3.883M	72.38±1.21
	OTTT	RSNN	3.883M	70.19±0.61
	OSTL	RSNN	3.883M	76.15±0.95
	e-prop	RSNN	3.883M	76.06±1.56
	pp-prop	RSNN	3.883M	73.98±1.63
	NLT-prop (ours)	RSNN	3.883M	**76.83±1.08**
N-MNIST	BPTT	RSNN	1.975M	95.70±0.18
	OTTT	RSNN	1.975M	98.22±0.37
	OSTL	RSNN	1.975M	97.50±0.11
	e-prop	RSNN	1.975M	97.47±0.05
	pp-prop	RSNN	1.975M	**98.49±0.08**
	NLT-prop (ours)	RSNN	1.975M	98.26±0.27
CIFAR10-DVS	BPTT	VGG-F (sWS)	9.520M	64.78±1.42
	OTTT	VGG-F (sWS)	9.520M	68.86±2.17
	OSTL	VGG-F (sWS)	9.520M	68.94±0.69
	e-prop	VGG-F (sWS)	9.520M	68.52±0.72
	pp-prop	VGG-F (sWS)	9.520M	67.90±0.60
	NLT-prop (ours)	VGG-F (sWS)	9.520M	**69.06±0.80**
DVS Gesture	BPTT	VGG-F (sWS)	9.521M	**93.56±1.34**
	OTTT	VGG-F (sWS)	9.521M	92.80±0.89
	OSTL	VGG-F (sWS)	9.521M	92.73±1.50
	e-prop	VGG-F (sWS)	9.521M	92.58±1.02
	pp-prop	VGG-F (sWS)	9.521M	91.67±2.78
	NLT-prop (ours)	VGG-F (sWS)	9.521M	93.41±0.43
DMTS	BPTT	RSNN	0.223M	75.21±3.85
	OTTT	RSNN	0.223M	67.09±7.80
	OSTL	RSNN	0.223M	78.21±1.21
	e-prop	RSNN	0.223M	78.38±1.79
	pp-prop	RSNN	0.223M	73.14±1.06
	NLT-prop (ours)	RSNN	0.223M	**78.62±4.71**

These results agree with the balance between omitted paths and approximation error described in Section A.9. Each additional order retains more propagation paths through another compressed contribution. Its benefit depends on how well that contribution recovers the missing temporal credit. Increasing Q also adds a pair of eligibility factors per order, raising the storage cost within the linear representation. The ablation therefore supports retaining nonlocal propagation and selecting Q according to the task and backbone, with low orders offering an economical starting point.

4.6 Space Complexity of Eligibility-Trace Storage

Table 3 compares the eligibility-trace storage of the online methods. For a dense recurrent layer with N hidden neurons and M input features, NLT-prop stores N local postsynaptic factors, $N+M$ presynaptic factors, and two length-N vectors for each of the Q nonlocal orders. The resulting trace count is $(2Q+2)N+M$, giving linear storage in N at fixed Q when $M = O(N)$. This representation retains nonlocal temporal contributions without storing a separate trace for every synapse. The trace storage is independent of sequence length. The corresponding derivations for dense layers and VGG-F are provided in Appendix A.8.3.

Table 2: Ablation of the nonlocal order Q on SHD with RSNN and DVS Gesture with VGG-F (sWS). ACC is reported as mean $\pm$ sample standard deviation in percent over five runs. The highest mean accuracy in each column is shown in **bold**, and the second-highest is underlined.

Q	RSNN	VGG-F (sWS)
0	75.97±0.97	92.27±0.63
1	**76.83±1.08**	93.33±1.91
2	76.08±1.83	92.35±1.32
3	75.11±3.34	93.18±0.89
4	72.42±6.19	92.27±0.91
5	75.84±3.05	93.26±1.18
6	71.59±5.18	93.41±1.12
7	75.95±1.68	**93.48±0.86**
8	71.37±5.51	91.67±1.10
9	76.51±0.83	93.33±1.16

Table 3: Theoretical eligibility-trace storage per sample for a dense recurrent layer with N hidden neurons and $M = O(N)$ input features. Q is the retained nonlocal order.

Method	**Eligibility-trace storage**
OTTT (Xiao et al., 2022)	$O(N)$
OSTL (Bohnstingl et al., 2023)	$O(N^2)$
Symmetric e-prop (Bellec et al., 2020)	$O(N^2)$
pp-prop (Wang et al., 2026)	$O(N)$
NLT-prop (ours)	$O((Q+1)N)$

Figure 2 compares the measured storage as network width increases. On both RSNN and VGG-F, NLT-prop exhibits linear growth at fixed Q, while the synapse-specific e-prop and OSTL traces exhibit quadratic growth under the tested width scaling. At width 1024, NLT-prop uses 0.053 MiB on SHD and 17.82 MiB on DVS Gesture, including auxiliary trace state, compared with 14.73 MiB and 13,833 MiB, respectively, for e-prop and OSTL. These results show that retaining nonlocal contributions with NLT-prop preserves linear trace-storage scaling across both backbones.

5 Conclusion

We introduced NLT-prop, an online learning method for recurrent SNNs that preserves nonlocal temporal credit through a compact eligibility representation. By separating local neuronal dynamics from cross-neuron propagation, NLT-prop combines separable local traces with deterministic low-rank approximations of successive nonlocal contributions, enabling forward gradient updates with linear eligibility-trace storage at fixed Q. Across five datasets with RSNN and VGG-F backbones, NLT-prop achieves the highest mean accuracy on three datasets and ranks second on the remaining two, exceeding BPTT in mean accuracy on four. Storage measurements further confirm linear growth with network width. These results demonstrate the value of retaining cross-neuron temporal information for accurate online learning under a limited eligibility-storage budget. Low-rank compression introduces approximation error, however, and the order ablations show that increasing Q does not consistently improve accuracy. Selecting Q therefore requires balancing recovered temporal credit against approximation error. The Jacobian-vector products and iterative factorization also add computational overhead, motivating more efficient approximations and adaptive order selection in future work.

AI use statement

In this work, we used generative AI tools for assisting in the writing of proofs, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis. We have not used

generative AI tools for help develop theoretical models or conceptual frameworks, implement methods, formulate mathematical claims, provide critical ingredients for proving mathematical claims, interpret results and Generate synthetic data sets are not applicable to this work. Additionally, we used generative AI tools for create or edit software code, draft parts of a research paper, summarize or analyze existing literature, sourcing/searching for information, edit a research paper to improve readability. We have reviewed all AI-assisted work. LLM-generated code was verified and tested by human, And We verified that the paragraphs revised by the LLM were consistent with the article's logic and experimental data. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Ethics statement

We agree to and comply with the ICLR Code of Ethics. We have ensured that all experiments were conducted in accordance with relevant ethical requirements and do not involve any infringement of individual privacy. All datasets used in this study are publicly available datasets. We have included executable source code in the supplementary materials, and all code will be released publicly on GitHub upon acceptance of the paper.

Reproducibility statement

We confirm that our work is highly reproducible. We have submitted executable source code as supplementary material, and all code will be made publicly available upon acceptance of the paper. In addition, we provide detailed mathematical derivations and proofs for the theoretical components of the work in the appendix.

References

Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo Di Nolfo, Tapan Nayak, Alexander Andreopoulos, Guillaume Garreau, Marcela Mendoza, Jeff Kusnitz, Michael Debole, Steve Esser, Tobi Delbruck, Myron Flickner, and Dharmendra Modha. A low power, fully event-based gesture recognition system. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7388–7397, 2017.

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*, 11:3625, 2020.

Gabriel Béna, Timo Wunderlich, Mahmoud Akl, Bernhard Vogginger, Christian Mayr, and Hector A. Gonzalez. Event-based backpropagation on the neuromorphic platform SpiNNaker2. In *2025 Neuro Inspired Computational Elements (NICE)*, pp. 1–10. IEEE, 2025.

Frederik Benzing, Marcelo Matheus Gauy, Asier Mujika, Anders Martinsson, and Angelika Steger. Optimal Kronecker-sum approximation of real time recurrent learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 604–613. PMLR, 2019.

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.

Benjamin Cramer, Yannik Stradmann, Johannes Schemmel, and Friedemann Zenke. The Heidelberg spiking data sets for the systematic evaluation of spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 33(7):2744–2757, 2022.

Mike Davies, Andreas Wild, Garrick Orchard, Yulia Sandamirskaya, Gabriel A. Fonseca Guerra, Prasad Joshi, Philipp Plank, and Sumedh R. Risbud. Advancing neuromorphic computing with Loihi: A survey of results and outlook. *Proceedings of the IEEE*, 109(5):911–934, 2021.

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

Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

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.

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.

Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 4 edition, 2013.

Wenzhe Guo, Mohammed E. Fouda, Ahmed M. Eltawil, and Khaled Nabil Salama. Efficient training of spiking neural networks with temporally-truncated local backpropagation through time. *Frontiers in Neuroscience*, 17:1047008, 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.

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 of *Proceedings of Machine Learning Research*, pp. 21806–21823. PMLR, 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 of *Proceedings of Machine Learning Research*, pp. 5189–5200. PMLR, 2021.

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, Yufei Guo, Shanghang Zhang, Shikuang Deng, Yongqing Hai, and Shi Gu. Differentiable spike: Rethinking gradient-descent for training spiking neural networks. In *Advances in Neural Information Processing Systems*, volume 34, pp. 23426–23439, 2021.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

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

Owen Marschall, Kyunghyun Cho, and Cristina Savin. A unified framework of online learning algorithms for training recurrent neural networks. *Journal of Machine Learning Research*, 21 (135):1–34, 2020.

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.

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.

James M. Murray. Local online learning in recurrent networks with random feedback. *eLife*, 8: e43299, 2019.

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

Garrick Orchard, Ajinkya Jayawant, Gregory K. Cohen, and Nitish Thakor. Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in Neuroscience*, 9:437, 2015.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 1310–1318. PMLR, 2013.

Nicolas Perez-Nieves and Dan F. M. Goodman. Sparse spiking gradient descent. In *Advances in Neural Information Processing Systems*, volume 34, pp. 11795–11808, 2021.

Michael Pfeiffer and Thomas Pfeil. Deep learning with spiking neurons: Opportunities and challenges. *Frontiers in Neuroscience*, 12:774, 2018.

Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575:607–617, 2019.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

Thomas M. Summe, Clemens JS Schaefer, and Siddharth Joshi. Estimating post-synaptic effects for online training of feed-forward SNNs. In *2024 International Conference on Neuromorphic Systems (ICONS)*, pp. 264–271. IEEE, 2024.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.

Corentin Tallec and Yann Ollivier. Unbiased online recurrent optimization. In *International Conference on Learning Representations*, 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.

Timo C. Wunderlich and Christian Pehle. Event-based backpropagation can compute exact gradients for spiking neural networks. *Scientific Reports*, 11:12829, 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.

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.

Friedemann Zenke and Surya Ganguli. SuperSpike: Supervised learning in multilayer spiking neural networks. *Neural Computation*, 30(6):1514–1541, 2018.

Friedemann Zenke and Tim P. Vogels. The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks. *Neural Computation*, 33(4):899–925, 2021.

A Derivation of NLT-prop

A.1 Discretization and Surrogate Sensitivities

Taking the resting potential as zero and absorbing membrane resistance into the current gives the subthreshold LIF equation (Gerstner et al., 2014). Writing the unscaled recurrent and input weights as $\overline{\boldsymbol{W}}$ and $\overline{\boldsymbol{V}}$, a forward Euler step gives

$$\tau_m \frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} = -\boldsymbol{u} + \boldsymbol{i}(t), \tag{9}$$

$$\boldsymbol{i}^t = \overline{\boldsymbol{W}}\boldsymbol{s}^{t-1} + \overline{\boldsymbol{V}}\boldsymbol{x}^t, \tag{10}$$

$$\boldsymbol{u}^t = \boldsymbol{u}^{t-1} + \frac{\Delta t}{\tau_m}(-\boldsymbol{u}^{t-1} + \boldsymbol{i}^t) - \vartheta \boldsymbol{s}^{t-1} \tag{11}$$

$$= \left(1 - \frac{\Delta t}{\tau_m}\right)\boldsymbol{u}^{t-1} + \frac{\Delta t}{\tau_m}\overline{\boldsymbol{W}}\boldsymbol{s}^{t-1} + \frac{\Delta t}{\tau_m}\overline{\boldsymbol{V}}\boldsymbol{x}^t - \vartheta \boldsymbol{s}^{t-1} \tag{12}$$

$$= \lambda \boldsymbol{u}^{t-1} + \boldsymbol{W}\boldsymbol{s}^{t-1} + \boldsymbol{V}\boldsymbol{x}^t - \vartheta \boldsymbol{s}^{t-1}, \tag{13}$$

where $\lambda = 1 - \Delta t/\tau_m$, $\boldsymbol{W} = (\Delta t/\tau_m)\overline{\boldsymbol{W}}$, and $\boldsymbol{V} = (\Delta t/\tau_m)\overline{\boldsymbol{V}}$. The state is the potential before the current threshold decision, and $\boldsymbol{s}^t = H(\boldsymbol{u}^t - \vartheta \mathbf{1})$. The derivative of this threshold operation is replaced by $\partial s_i^t/\partial u_j^t = \delta_{ij}\psi_i^t$, with the triangular surrogate in Eq. (2). Consequently,

$$u_k^t = \lambda u_k^{t-1} + \sum_h W_{kh} s_h^{t-1} + \sum_j V_{kj} x_j^t - \vartheta s_k^{t-1}, \tag{14}$$

$$\frac{\partial u_k^t}{\partial u_h^{t-1}} = \lambda \delta_{kh} + \sum_a W_{ka} \frac{\partial s_a^{t-1}}{\partial u_h^{t-1}} - \vartheta \frac{\partial s_k^{t-1}}{\partial u_h^{t-1}} \tag{15}$$

$$= \lambda \delta_{kh} + \sum_a W_{ka}\delta_{ah}\psi_a^{t-1} - \vartheta \delta_{kh}\psi_k^{t-1} \tag{16}$$

$$= \lambda \delta_{kh} + (W_{kh} - \vartheta \delta_{kh})\psi_h^{t-1}. \tag{17}$$

The following identities describe surrogate sensitivities along a trajectory with fixed parameters. Applying the chain rule to Eq. (1) gives

$$\frac{\partial u_k^t}{\partial u_h^{t-1}} = \lambda \delta_{kh} + (W_{kh} - \vartheta \delta_{kh})\psi_h^{t-1}. \tag{18}$$

For $\boldsymbol{\Theta} = [\boldsymbol{W}\ \boldsymbol{V}]$, differentiation at fixed previous state gives $\partial u_k^t/\partial \Theta_{ij} = \delta_{ki}\zeta_j^t$. Including the dependence through previous states yields

$$E_{k,(i,j)}^t = \sum_h J_{kh}^t E_{h,(i,j)}^{t-1} + \delta_{ki}\zeta_j^t, \qquad \boldsymbol{E}^0 = \mathbf{0}. \tag{19}$$

Here and below, a zero matrix has the dimensions specified by its context. Contracting this recursion with $\boldsymbol{g}^t$ gives the surrogate gradient for the current loss. Online NLT-prop evaluates the approximate recursion using the current weights and stored factors at every step.

To derive the sensitivity recursion, we distinguish the direct derivative at a fixed previous state from the total sensitivity. With $\boldsymbol{\Theta} = [\boldsymbol{W}\ \boldsymbol{V}] \in \mathbb{R}^{N\times(N+M)}$ and $\boldsymbol{\zeta}^t = [(\boldsymbol{s}^{t-1})^\top\ (\boldsymbol{x}^t)^\top]^\top$,

$$u_k^t = \lambda u_k^{t-1} + \sum_{a=1}^{N+M} \Theta_{ka}\zeta_a^t - \vartheta s_k^{t-1}, \tag{20}$$

$$B_{k,(i,j)}^t = \left.\frac{\partial u_k^t}{\partial \Theta_{ij}}\right|_{\boldsymbol{u}^{t-1},\boldsymbol{x}^t} = \sum_a \delta_{ki}\delta_{aj}\zeta_a^t = \delta_{ki}\zeta_j^t, \tag{21}$$

$$E_{k,(i,j)}^t = B_{k,(i,j)}^t + \sum_h \frac{\partial u_k^t}{\partial u_h^{t-1}}\frac{\partial u_h^{t-1}}{\partial \Theta_{ij}} \tag{22}$$

$$= \delta_{ki}\zeta_j^t + \sum_h J_{kh}^t E_{h,(i,j)}^{t-1}. \tag{23}$$

Thus $\boldsymbol{B}^t, \boldsymbol{E}^t \in \mathbb{R}^{N\times N(N+M)}$. The learning signal contracts the state index:

$$\frac{\partial \mathcal{L}^t}{\partial \Theta_{ij}} = \sum_k \frac{\partial \mathcal{L}^t}{\partial u_k^t}\frac{\partial u_k^t}{\partial \Theta_{ij}} = \sum_k g_k^t E_{k,(i,j)}^t. \tag{24}$$

A.2 Diagonal and Off-Diagonal Components

Let $\boldsymbol{W}_{\mathrm{d}} = \mathrm{Diag}(\mathrm{diag}(\boldsymbol{W}))$. Substituting $\boldsymbol{W} = \boldsymbol{W}_{\mathrm{d}} + (\boldsymbol{W} - \boldsymbol{W}_{\mathrm{d}})$ into the temporal Jacobian yields

$$\boldsymbol{J}^t = \lambda \boldsymbol{I} + (\boldsymbol{W} - \vartheta \boldsymbol{I})\boldsymbol{\Psi}^{t-1} \tag{25}$$

$$= \lambda \boldsymbol{I} + (\boldsymbol{W}_{\mathrm{d}} - \vartheta \boldsymbol{I})\boldsymbol{\Psi}^{t-1} + (\boldsymbol{W} - \boldsymbol{W}_{\mathrm{d}})\boldsymbol{\Psi}^{t-1} \tag{26}$$

$$= \underbrace{\mathrm{Diag}\big(\lambda \mathbf{1} + (\mathrm{diag}(\boldsymbol{W}) - \vartheta \mathbf{1}) \odot \boldsymbol{\psi}^{t-1}\big)}_{\boldsymbol{D}^t} + \underbrace{\mathrm{offdiag}(\boldsymbol{W})\boldsymbol{\Psi}^{t-1}}_{\boldsymbol{C}^t}. \tag{27}$$

Entrywise, $D_{kh}^t = \delta_{kh} d_k^t$, $d_k^t = \lambda + (W_{kk} - \vartheta)\psi_k^{t-1}$, and $C_{kh}^t = (1-\delta_{kh})W_{kh}\psi_h^{t-1}$. The diagonal retains the leak, reset derivative, and recurrent self-connection; the off-diagonal contains the cross-neuron transitions.

A.3 Expansion by Nonlocal Transition Order

The first three substitutions make the temporal ordering explicit:

$$\boldsymbol{E}^1 = \boldsymbol{B}^1, \tag{28}$$

$$\boldsymbol{E}^2 = \boldsymbol{J}^2\boldsymbol{E}^1 + \boldsymbol{B}^2 = \boldsymbol{J}^2\boldsymbol{B}^1 + \boldsymbol{B}^2, \tag{29}$$

$$\boldsymbol{E}^3 = \boldsymbol{J}^3(\boldsymbol{J}^2\boldsymbol{B}^1 + \boldsymbol{B}^2) + \boldsymbol{B}^3 = \boldsymbol{J}^3\boldsymbol{J}^2\boldsymbol{B}^1 + \boldsymbol{J}^3\boldsymbol{B}^2 + \boldsymbol{B}^3, \tag{30}$$

$$\boldsymbol{J}^3\boldsymbol{J}^2\boldsymbol{B}^1 = (\boldsymbol{D}^3 + \boldsymbol{C}^3)(\boldsymbol{D}^2 + \boldsymbol{C}^2)\boldsymbol{B}^1 \tag{31}$$

$$= \boldsymbol{D}^3\boldsymbol{D}^2\boldsymbol{B}^1 + \boldsymbol{D}^3\boldsymbol{C}^2\boldsymbol{B}^1 + \boldsymbol{C}^3\boldsymbol{D}^2\boldsymbol{B}^1 + \boldsymbol{C}^3\boldsymbol{C}^2\boldsymbol{B}^1. \tag{32}$$

The four terms contain zero, one, one, and two nonlocal transitions, respectively. Their multiplication order follows the time indices.

Repeated substitution in Eq. (3) gives

$$\boldsymbol{E}^t = \sum_{\tau=1}^{t} \boldsymbol{J}^t\boldsymbol{J}^{t-1}\cdots\boldsymbol{J}^{\tau+1}\boldsymbol{B}^\tau, \tag{33}$$

where the empty product for $\tau = t$ is the identity. Substitute $\boldsymbol{J}^s = \boldsymbol{D}^s + \boldsymbol{C}^s$ into each ordered product. Grouping its terms by the number of $\boldsymbol{C}$ factors defines $\boldsymbol{E}_\nu^t$. This construction uses the distributive law and preserves the order of the generally noncommuting matrices. The resulting recursions are

$$\begin{aligned} \boldsymbol{E}_0^t &= \boldsymbol{D}^t\boldsymbol{E}_0^{t-1} + \boldsymbol{B}^t, \\ \boldsymbol{E}_\nu^t &= \boldsymbol{D}^t\boldsymbol{E}_\nu^{t-1} + \boldsymbol{C}^t\boldsymbol{E}_{\nu-1}^{t-1}, \qquad \nu \geq 1, \\ \boldsymbol{E}_\nu^0 &= \mathbf{0}, \qquad \boldsymbol{E}^t = \sum_{\nu=0}^{t-1} \boldsymbol{E}_\nu^t. \end{aligned} \tag{34}$$

The first term preserves the current order through a local transition, and the second increments the order through a cross-neuron transition. Induction on t proves the final equality. At $t = 1$, only $\boldsymbol{B}^1$ is present. At each later step, multiplication by $\boldsymbol{D}^t + \boldsymbol{C}^t$ accounts for every continuation once, and $\boldsymbol{B}^t$ supplies the new order-zero terms.

A.4 Local Recursion and Separable Approximation

Starting from $E^0_{0,k,(i,j)} = 0$, diagonal propagation preserves the form $E^t_{0,k,(i,j)} = \delta_{ki} Z^t_{ij}$. Substituting this form into the order-zero recursion gives

$$
\begin{aligned}
E^t_{0,k,(i,j)} &= \sum_h D^t_{kh} E^{t-1}_{0,h,(i,j)} + \delta_{ki}\zeta^t_j && (35)\\
&= \sum_h \delta_{kh} d^t_k \delta_{hi} Z^{t-1}_{ij} + \delta_{ki}\zeta^t_j && (36)\\
&= \delta_{ki}(d^t_i Z^{t-1}_{ij} + \zeta^t_j), && (37)\\
Z^t_{ij} &= d^t_i Z^{t-1}_{ij} + \zeta^t_j && (38)\\
&= d^t_i (d^{t-1}_i Z^{t-2}_{ij} + \zeta^{t-1}_j) + \zeta^t_j && (39)\\
&= \sum_{\tau=1}^{t} \left(\prod_{s=\tau+1}^{t} d^s_i \right) \zeta^\tau_j . && (40)
\end{aligned}
$$

In matrix form, $\boldsymbol{Z}^t = \boldsymbol{D}^t \boldsymbol{Z}^{t-1} + \mathbf{1}(\boldsymbol{\zeta}^t)^\top$. Substitute the stored approximation $\widehat{\boldsymbol{Z}}^{t-1} = \boldsymbol{\ell}^{t-1}(\boldsymbol{p}^{t-1})^\top$ into this update:

$$
\begin{aligned}
\boldsymbol{Z}^t_{\text{cand}} &= \boldsymbol{D}^t \widehat{\boldsymbol{Z}}^{t-1} + \mathbf{1}(\boldsymbol{\zeta}^t)^\top && (41)\\
&= \boldsymbol{D}^t \boldsymbol{\ell}^{t-1}(\boldsymbol{p}^{t-1})^\top + \mathbf{1}(\boldsymbol{\zeta}^t)^\top && (42)\\
&= (\boldsymbol{d}^t \odot \boldsymbol{\ell}^{t-1})(\boldsymbol{p}^{t-1})^\top + \mathbf{1}(\boldsymbol{\zeta}^t)^\top, && (43)\\
\widehat{\boldsymbol{Z}}^t &= \mathcal{P}_1[\boldsymbol{Z}^t_{\text{cand}}] = \boldsymbol{\ell}^t(\boldsymbol{p}^t)^\top, && (44)\\
\widehat{E}^t_{0,k,(i,j)} &= \delta_{ki}\widehat{Z}^t_{ij} = \delta_{ki}\ell^t_i p^t_j. && (45)
\end{aligned}
$$

The candidate has rank at most two. Its projection stores $N+(N+M)$ factors instead of $N(N+M)$ local trace entries. Both recurrent and input traces use the same postsynaptic factor because their presynaptic activities are concatenated before compression.

A.5 Compression of Two Outer Products

Let $\boldsymbol{M} = \boldsymbol{x}_1\boldsymbol{y}_1^\top + \boldsymbol{x}_2\boldsymbol{y}_2^\top = \boldsymbol{X}\boldsymbol{Y}^\top$, with $\boldsymbol{X} = [\boldsymbol{x}_1\ \boldsymbol{x}_2]$ and $\boldsymbol{Y} = [\boldsymbol{y}_1\ \boldsymbol{y}_2]$. The Eckart–Young theorem states that the leading singular component solves

$$
\mathcal{P}_1[\boldsymbol{M}] = \operatorname*{argmin}_{\operatorname{rank}(\boldsymbol{A})\le 1} \|\boldsymbol{M} - \boldsymbol{A}\|_F^2 = \sigma_1 \boldsymbol{u}_1 \boldsymbol{v}_1^\top . \tag{46}
$$

Its squared residual is $\sum_{j\ge 2}\sigma_j^2$ (Eckart & Young, 1936). For the rank-two candidate, the singular component follows from a two-dimensional eigenproblem. Swap the two outer products if necessary so that $\|\boldsymbol{x}_1\|_2 \ge \|\boldsymbol{x}_2\|_2$. Gram–Schmidt orthogonalization (Golub & Van Loan, 2013) gives

$$
\begin{aligned}
r_{11} &= \|\boldsymbol{x}_1\|_2, & \boldsymbol{u}_1 &= \boldsymbol{x}_1/r_{11}, && (47)\\
r_{12} &= \boldsymbol{u}_1^\top \boldsymbol{x}_2, & \widetilde{\boldsymbol{u}}_2 &= \boldsymbol{x}_2 - r_{12}\boldsymbol{u}_1, && (48)\\
r_{22} &= \|\widetilde{\boldsymbol{u}}_2\|_2, & \boldsymbol{u}_2 &= \widetilde{\boldsymbol{u}}_2/r_{22}, && (49)\\
\boldsymbol{X} &= \boldsymbol{U}\boldsymbol{R}_x, & \boldsymbol{R}_x &= \begin{bmatrix} r_{11} & r_{12} \\ 0 & r_{22} \end{bmatrix}, && (50)
\end{aligned}
$$

where $\boldsymbol{U} = [\boldsymbol{u}_1\ \boldsymbol{u}_2]$. Define $g_{ab} = \boldsymbol{y}_a^\top \boldsymbol{y}_b$. Substitution yields

$$\boldsymbol{M}\boldsymbol{M}^\top = (\boldsymbol{U}\boldsymbol{R}_x\boldsymbol{Y}^\top)(\boldsymbol{Y}\boldsymbol{R}_x^\top\boldsymbol{U}^\top) \tag{51}$$

$$= \boldsymbol{U}\underbrace{\boldsymbol{R}_x(\boldsymbol{Y}^\top\boldsymbol{Y})\boldsymbol{R}_x^\top}_{\boldsymbol{S}}\boldsymbol{U}^\top, \tag{52}$$

$$s_{11} = r_{11}^2 g_{11} + 2r_{11}r_{12}g_{12} + r_{12}^2 g_{22}, \tag{53}$$

$$s_{12} = r_{22}(r_{11}g_{12} + r_{12}g_{22}), \qquad s_{22} = r_{22}^2 g_{22}. \tag{54}$$

The eigenvalues satisfy

$$0 = \det(\boldsymbol{S} - \rho\boldsymbol{I}) = (s_{11} - \rho)(s_{22} - \rho) - s_{12}^2 \tag{55}$$

$$= \rho^2 - (s_{11} + s_{22})\rho + s_{11}s_{22} - s_{12}^2, \tag{56}$$

$$\rho_+ = \frac{s_{11} + s_{22} + \sqrt{(s_{11} - s_{22})^2 + 4s_{12}^2}}{2}. \tag{57}$$

A leading eigenvector is obtained by normalizing the longer of $(s_{12}, \rho_+ - s_{11})^\top$ and $(\rho_+ - s_{22}, s_{12})^\top$. Denote it by $\boldsymbol{w}$. If both candidates vanish, choose $(1, 0)^\top$. The corresponding singular factors follow by direct substitution:

$$\boldsymbol{u} = \boldsymbol{U}\boldsymbol{w}, \tag{58}$$

$$\widetilde{\boldsymbol{v}} = \boldsymbol{M}^\top\boldsymbol{u} = (\boldsymbol{y}_1\boldsymbol{x}_1^\top + \boldsymbol{y}_2\boldsymbol{x}_2^\top)\boldsymbol{u} = \boldsymbol{y}_1(\boldsymbol{x}_1^\top\boldsymbol{u}) + \boldsymbol{y}_2(\boldsymbol{x}_2^\top\boldsymbol{u}), \tag{59}$$

$$\sigma = \|\widetilde{\boldsymbol{v}}\|_2, \qquad \boldsymbol{v} = \widetilde{\boldsymbol{v}}/\sigma, \tag{60}$$

$$\boldsymbol{x} = \sqrt{\sigma}\boldsymbol{u}, \qquad \boldsymbol{y} = \sqrt{\sigma}\boldsymbol{v}, \tag{61}$$

$$\boldsymbol{x}\boldsymbol{y}^\top = \sigma\boldsymbol{u}\boldsymbol{v}^\top = \mathcal{P}_1[\boldsymbol{M}]. \tag{62}$$

A zero candidate gives zero factors. When the column space is one-dimensional, the second basis vector is set to zero and the construction reduces to the first direction. The factor signs are changed together, when necessary, so that the largest-magnitude entry of the right factor is nonnegative.

For an operator available through products with vectors, a finite number of alternating power iterations supplies an approximate leading singular component (Golub & Van Loan, 2013). Given a normalized right direction $\boldsymbol{v}_0$, iteration j evaluates

$$\widetilde{\boldsymbol{u}}_j = \boldsymbol{M}\boldsymbol{v}_{j-1}, \qquad \boldsymbol{u}_j = \widetilde{\boldsymbol{u}}_j/\|\widetilde{\boldsymbol{u}}_j\|_2, \tag{63}$$

$$\widetilde{\boldsymbol{v}}_j = \boldsymbol{M}^\top\boldsymbol{u}_j, \qquad \boldsymbol{v}_j = \widetilde{\boldsymbol{v}}_j/\|\widetilde{\boldsymbol{v}}_j\|_2, \tag{64}$$

$$\widetilde{\boldsymbol{v}}_j = \frac{\boldsymbol{M}^\top\boldsymbol{M}\boldsymbol{v}_{j-1}}{\|\boldsymbol{M}\boldsymbol{v}_{j-1}\|_2}. \tag{65}$$

After K iterations, set $\boldsymbol{w} = \boldsymbol{M}\boldsymbol{v}_K$, $\sigma = \|\boldsymbol{w}\|_2$, and return $\boldsymbol{x} = \sqrt{\sigma}\boldsymbol{w}/\sigma$, $\boldsymbol{y} = \sqrt{\sigma}\boldsymbol{v}_K$. This operation is denoted by $\mathcal{P}_1^{(K)}$. For a sum of two outer products, the required products are

$$\boldsymbol{M}\boldsymbol{v} = \boldsymbol{x}_1(\boldsymbol{y}_1^\top\boldsymbol{v}) + \boldsymbol{x}_2(\boldsymbol{y}_2^\top\boldsymbol{v}), \qquad \boldsymbol{M}^\top\boldsymbol{u} = \boldsymbol{y}_1(\boldsymbol{x}_1^\top\boldsymbol{u}) + \boldsymbol{y}_2(\boldsymbol{x}_2^\top\boldsymbol{u}). \tag{66}$$

VGG-F uses this iteration for its trace compressions, initializing from the right factor with the larger norm. RSNN uses the closed-form two-dimensional construction for sums of outer products. Denominators use the configured positive floor, and a vanishing operator product returns zero factors or preserves the previous warm-start direction as appropriate.

A.6 Nonlocal Operators and Shared Presynaptic Factors

For the lateral model, products with the nonlocal Jacobian and its transpose are

$$\begin{aligned}\boldsymbol{C}^t\boldsymbol{v} &= \boldsymbol{W}(\boldsymbol{\psi}^{t-1} \odot \boldsymbol{v}) - \operatorname{diag}(\boldsymbol{W}) \odot \boldsymbol{\psi}^{t-1} \odot \boldsymbol{v},\\ (\boldsymbol{C}^t)^\top\boldsymbol{v} &= \boldsymbol{\psi}^{t-1} \odot \big(\boldsymbol{W}^\top\boldsymbol{v} - \operatorname{diag}(\boldsymbol{W}) \odot \boldsymbol{v}\big).\end{aligned} \tag{67}$$

These products follow by distributing the diagonal matrices:

$$\boldsymbol{C}^t\boldsymbol{v} = (\boldsymbol{W} - \boldsymbol{W}_\mathrm{d})\boldsymbol{\Psi}^{t-1}\boldsymbol{v} = \boldsymbol{W}(\boldsymbol{\psi}^{t-1} \odot \boldsymbol{v}) - \operatorname{diag}(\boldsymbol{W}) \odot \boldsymbol{\psi}^{t-1} \odot \boldsymbol{v}, \tag{68}$$

$$(\boldsymbol{C}^t)^\top\boldsymbol{v} = \boldsymbol{\Psi}^{t-1}(\boldsymbol{W}^\top - \boldsymbol{W}_\mathrm{d})\boldsymbol{v} = \boldsymbol{\psi}^{t-1} \odot (\boldsymbol{W}^\top\boldsymbol{v} - \operatorname{diag}(\boldsymbol{W}) \odot \boldsymbol{v}). \tag{69}$$

Using these products in I power iterations gives balanced factors $\boldsymbol{a}^t, \boldsymbol{b}^t$. Removing their diagonal gives

$$\widetilde{\boldsymbol{C}}^t = \mathrm{offdiag}(\boldsymbol{a}^t(\boldsymbol{b}^t)^\top) \tag{70}$$

$$= \boldsymbol{a}^t(\boldsymbol{b}^t)^\top - \mathrm{Diag}(\boldsymbol{a}^t \odot \boldsymbol{b}^t), \tag{71}$$

$$\widetilde{\boldsymbol{C}}^t\boldsymbol{v} = \boldsymbol{a}^t((\boldsymbol{b}^t)^\top\boldsymbol{v}) - \boldsymbol{a}^t \odot \boldsymbol{b}^t \odot \boldsymbol{v}. \tag{72}$$

The masked operator is represented by two vectors and an implicit diagonal subtraction; its matrix rank can exceed one after masking.

Here $\mathrm{diag}(\boldsymbol{W})$ extracts a vector, whereas Diag constructs a diagonal matrix. Power iteration starts from the normalized all-ones vector at the beginning of a sequence and reuses the preceding right direction at subsequent steps. For the balanced factors $\boldsymbol{a}^t, \boldsymbol{b}^t$, diagonal masking gives

$$\widetilde{\boldsymbol{C}}^t\boldsymbol{v} = \boldsymbol{a}^t\big((\boldsymbol{b}^t)^\top\boldsymbol{v}\big) - \boldsymbol{a}^t \odot \boldsymbol{b}^t \odot \boldsymbol{v}. \tag{73}$$

The second term removes the self-transition introduced by the outer product. It costs $O(N)$ operations and requires no matrix storage.

For the first-order source, write $\boldsymbol{w} = \boldsymbol{b}^t \odot \boldsymbol{\ell}^{t-1}$. Then $\boldsymbol{S} = \widetilde{\boldsymbol{C}}^t\,\mathrm{Diag}(\boldsymbol{\ell}^{t-1}) = \boldsymbol{a}^t\boldsymbol{w}^\top - \mathrm{Diag}(\boldsymbol{a}^t \odot \boldsymbol{w})$, and

$$\begin{aligned} \boldsymbol{S}\boldsymbol{v} &= \boldsymbol{a}^t(\boldsymbol{w}^\top\boldsymbol{v}) - \boldsymbol{a}^t \odot \boldsymbol{w} \odot \boldsymbol{v}, \\ \boldsymbol{S}^\top\boldsymbol{v} &= \boldsymbol{w}((\boldsymbol{a}^t)^\top\boldsymbol{v}) - \boldsymbol{a}^t \odot \boldsymbol{w} \odot \boldsymbol{v}. \end{aligned} \tag{74}$$

Power iteration initialized from $\mathrm{normalize}(\boldsymbol{w})$ supplies the rank-one source $\boldsymbol{S}_1^t$. For $\nu > 1$, the source is already an outer product because $\widetilde{\boldsymbol{C}}^t\widehat{\boldsymbol{H}}_{\nu-1}^{t-1} = (\widetilde{\boldsymbol{C}}^t\boldsymbol{r}_{\nu-1}^{t-1})(\boldsymbol{q}_{\nu-1}^{t-1})^\top$. All orders are updated from the factors at $t-1$ before the new factors replace them.

The first-order source follows from the local neuronal factor $\widehat{\boldsymbol{H}}_0^{t-1} = \mathrm{Diag}(\boldsymbol{\ell}^{t-1})$:

$$\widetilde{\boldsymbol{C}}^t\widehat{\boldsymbol{H}}_0^{t-1} = [\boldsymbol{a}^t(\boldsymbol{b}^t)^\top - \mathrm{Diag}(\boldsymbol{a}^t \odot \boldsymbol{b}^t)]\,\mathrm{Diag}(\boldsymbol{\ell}^{t-1}) \tag{75}$$

$$= \boldsymbol{a}^t(\boldsymbol{b}^t \odot \boldsymbol{\ell}^{t-1})^\top - \mathrm{Diag}(\boldsymbol{a}^t \odot \boldsymbol{b}^t \odot \boldsymbol{\ell}^{t-1}) \tag{76}$$

$$= \boldsymbol{a}^t\boldsymbol{w}^\top - \mathrm{Diag}(\boldsymbol{a}^t \odot \boldsymbol{w}), \qquad \boldsymbol{w} = \boldsymbol{b}^t \odot \boldsymbol{\ell}^{t-1}. \tag{77}$$

Applying the power-iteration products in Eq. (74) gives $\boldsymbol{S}_1^t = \boldsymbol{x}_2\boldsymbol{y}_2^\top$. At a higher order, substitute $\widehat{\boldsymbol{H}}_{\nu-1}^{t-1} = \boldsymbol{r}_{\nu-1}^{t-1}(\boldsymbol{q}_{\nu-1}^{t-1})^\top$:

$$\widetilde{\boldsymbol{C}}^t\widehat{\boldsymbol{H}}_{\nu-1}^{t-1} = \widetilde{\boldsymbol{C}}^t\boldsymbol{r}_{\nu-1}^{t-1}(\boldsymbol{q}_{\nu-1}^{t-1})^\top \tag{78}$$

$$= \big[\boldsymbol{a}^t((\boldsymbol{b}^t)^\top\boldsymbol{r}_{\nu-1}^{t-1}) - \boldsymbol{a}^t \odot \boldsymbol{b}^t \odot \boldsymbol{r}_{\nu-1}^{t-1}\big]\,(\boldsymbol{q}_{\nu-1}^{t-1})^\top. \tag{79}$$

Thus $\boldsymbol{x}_2 = \widetilde{\boldsymbol{C}}^t\boldsymbol{r}_{\nu-1}^{t-1}$ and $\boldsymbol{y}_2 = \boldsymbol{q}_{\nu-1}^{t-1}$ for $\nu > 1$. The propagated old mode supplies the other outer product:

$$\boldsymbol{D}^t\widehat{\boldsymbol{H}}_\nu^{t-1} = \boldsymbol{D}^t\boldsymbol{r}_\nu^{t-1}(\boldsymbol{q}_\nu^{t-1})^\top = (\boldsymbol{d}^t \odot \boldsymbol{r}_\nu^{t-1})(\boldsymbol{q}_\nu^{t-1})^\top, \tag{80}$$

$$\boldsymbol{H}_{\nu,\mathrm{cand}}^t = (\boldsymbol{d}^t \odot \boldsymbol{r}_\nu^{t-1})(\boldsymbol{q}_\nu^{t-1})^\top + \boldsymbol{x}_2\boldsymbol{y}_2^\top, \tag{81}$$

$$\widehat{\boldsymbol{H}}_\nu^t = \mathcal{P}_1[\boldsymbol{H}_{\nu,\mathrm{cand}}^t] = \boldsymbol{r}_\nu^t(\boldsymbol{q}_\nu^t)^\top. \tag{82}$$

To express the shared presynaptic representation, define the linear embedding $\mathcal{T}_{\boldsymbol{p}}(\boldsymbol{H})_{k,(i,j)} = H_{ki}p_j$. Then $\widehat{\boldsymbol{E}}_0^t = \mathcal{T}_{\boldsymbol{p}^t}(\mathrm{Diag}(\boldsymbol{\ell}^t))$ and $\widehat{\boldsymbol{E}}_\nu^t = \mathcal{T}_{\boldsymbol{p}^t}(\widehat{\boldsymbol{H}}_\nu^t)$ for $\nu \geq 1$. Reusing $\boldsymbol{p}^t$ across orders is an additional factorization step. Its effect can be included explicitly in the recursion residual. Define $\widehat{\boldsymbol{H}}_0^t = \mathrm{Diag}(\boldsymbol{\ell}^t)$ and $\boldsymbol{M}_\nu^t = \boldsymbol{D}^t\widehat{\boldsymbol{H}}_\nu^{t-1} + \boldsymbol{C}^t\widehat{\boldsymbol{H}}_{\nu-1}^{t-1}$. For $\nu \geq 1$,

$$\begin{aligned} \boldsymbol{R}_\nu^t &:= \widehat{\boldsymbol{E}}_\nu^t - \boldsymbol{D}^t\widehat{\boldsymbol{E}}_\nu^{t-1} - \boldsymbol{C}^t\widehat{\boldsymbol{E}}_{\nu-1}^{t-1} \\ &= \mathcal{T}_{\boldsymbol{p}^t}(\widehat{\boldsymbol{H}}_\nu^t - \boldsymbol{M}_\nu^t) + \mathcal{T}_{\boldsymbol{p}^t - \boldsymbol{p}^{t-1}}(\boldsymbol{M}_\nu^t). \end{aligned} \tag{83}$$

To obtain Eq. (83), use linearity in both arguments of $\mathcal{T}$ and the identity $\boldsymbol{A}\mathcal{T}_{\boldsymbol{p}}(\boldsymbol{H}) = \mathcal{T}_{\boldsymbol{p}}(\boldsymbol{A}\boldsymbol{H})$:

$$\boldsymbol{R}_\nu^t = \mathcal{T}_{\boldsymbol{p}^t}(\widehat{\boldsymbol{H}}_\nu^t) - \mathcal{T}_{\boldsymbol{p}^{t-1}}(\boldsymbol{M}_\nu^t) \tag{84}$$

$$= \mathcal{T}_{\boldsymbol{p}^t}(\widehat{\boldsymbol{H}}_\nu^t) - \mathcal{T}_{\boldsymbol{p}^t}(\boldsymbol{M}_\nu^t) + \mathcal{T}_{\boldsymbol{p}^t}(\boldsymbol{M}_\nu^t) - \mathcal{T}_{\boldsymbol{p}^{t-1}}(\boldsymbol{M}_\nu^t) \tag{85}$$

$$= \mathcal{T}_{\boldsymbol{p}^t}(\widehat{\boldsymbol{H}}_\nu^t - \boldsymbol{M}_\nu^t) + \mathcal{T}_{\boldsymbol{p}^t - \boldsymbol{p}^{t-1}}(\boldsymbol{M}_\nu^t), \tag{86}$$

$$\|\mathcal{T}_{\boldsymbol{p}}(\boldsymbol{H})\|_F^2 = \sum_{k,i,j} H_{ki}^2 p_j^2 = \left(\sum_{k,i} H_{ki}^2\right)\left(\sum_j p_j^2\right) = \|\boldsymbol{H}\|_F^2 \|\boldsymbol{p}\|_2^2. \tag{87}$$

Define $\boldsymbol{F}_\nu^t = \widehat{\boldsymbol{H}}_\nu^t - \boldsymbol{H}_{\nu,\mathrm{cand}}^t$ and $\boldsymbol{U}_\nu^t = \boldsymbol{S}_\nu^t - \widetilde{\boldsymbol{C}}^t \widehat{\boldsymbol{H}}_{\nu-1}^{t-1}$. Expanding the candidate gives

$$\widehat{\boldsymbol{H}}_\nu^t - \boldsymbol{M}_\nu^t = \boldsymbol{F}_\nu^t + \boldsymbol{S}_\nu^t - \boldsymbol{C}^t \widehat{\boldsymbol{H}}_{\nu-1}^{t-1} \tag{88}$$

$$= \boldsymbol{F}_\nu^t + \boldsymbol{U}_\nu^t + (\widetilde{\boldsymbol{C}}^t - \boldsymbol{C}^t)\widehat{\boldsymbol{H}}_{\nu-1}^{t-1}, \tag{89}$$

$$\|\boldsymbol{R}_\nu^t\|_F \le \|\boldsymbol{p}^t\|_2 \big(\|\boldsymbol{F}_\nu^t\|_F + \|\boldsymbol{U}_\nu^t\|_F + \|\widetilde{\boldsymbol{C}}^t - \boldsymbol{C}^t\|_2 \|\widehat{\boldsymbol{H}}_{\nu-1}^{t-1}\|_F\big) + \|\boldsymbol{p}^t - \boldsymbol{p}^{t-1}\|_2 \|\boldsymbol{M}_\nu^t\|_F. \tag{90}$$

For $\nu > 1$, $\boldsymbol{U}_\nu^t = 0$ before factor clipping because the source is already an outer product. The first-order source includes its finite-iteration approximation residual.

The first term contains Jacobian approximation and neuronal-factor compression. The second captures the change in the shared presynaptic factor. Since $\|\mathcal{T}_{\boldsymbol{p}}(\boldsymbol{H})\|_F = \|\boldsymbol{p}\|_2 \|\boldsymbol{H}\|_F$, both contributions admit direct norm bounds. For the local order, define $\boldsymbol{R}_0^t = \widehat{\boldsymbol{E}}_0^t - \boldsymbol{D}^t \widehat{\boldsymbol{E}}_0^{t-1} - \boldsymbol{B}^t$.

The eligibility factors are initialized to zero. The evaluated configurations apply elementwise clipping with a magnitude limit of 20 after compression. Clipping is included in the residuals above.

A.7 Gradient Contraction and Forward Update Pseudocode

Substituting all stored factors into the gradient contraction gives every step of the parameter update:

$$\widehat{\nabla}_{\Theta_{ij}} \mathcal{L}^t = \sum_k g_k^t \widehat{E}_{k,(i,j)}^t \tag{91}$$

$$= \sum_k g_k^t \left(\delta_{ki} \ell_i^t + \sum_{\nu=1}^{Q} r_{\nu,k}^t q_{\nu,i}^t\right) p_j^t \tag{92}$$

$$= \left(g_i^t \ell_i^t + \sum_{\nu=1}^{Q} \sum_k g_k^t r_{\nu,k}^t q_{\nu,i}^t\right) p_j^t \tag{93}$$

$$= \left(g_i^t \ell_i^t + \sum_{\nu=1}^{Q} ((\boldsymbol{g}^t)^\top \boldsymbol{r}_\nu^t) q_{\nu,i}^t\right) p_j^t = z_i^t p_j^t, \tag{94}$$

$$\widehat{\nabla}_{\boldsymbol{\Theta}} \mathcal{L}^t = \boldsymbol{z}^t (\boldsymbol{p}^t)^\top, \tag{95}$$

$$\widehat{\nabla}_{\boldsymbol{W}} \mathcal{L}^t = \boldsymbol{z}^t (\boldsymbol{p}_{\mathrm{rec}}^t)^\top, \qquad \widehat{\nabla}_{\boldsymbol{V}} \mathcal{L}^t = \boldsymbol{z}^t (\boldsymbol{p}_{\mathrm{in}}^t)^\top. \tag{96}$$

The pseudocode below describes one RSNN layer. $\mathrm{Pair}(\boldsymbol{x}_1, \boldsymbol{y}_1, \boldsymbol{x}_2, \boldsymbol{y}_2)$ returns the balanced factors from Appendix A.5. $\mathrm{Power}(\boldsymbol{A}, I, \boldsymbol{v})$ applies I iterations through operator products and returns two balanced factors and the updated right direction. Clip applies the configured elementwise factor limit. For several layers, complete the current forward pass, update each layer's traces, obtain the current spatial learning signals, and apply the same contraction in each layer. Batch gradients are averaged over samples.

The old factors remain available throughout the order loop, so order ν reads order $\nu - 1$ at time $t - 1$. An update interval of one applies each available gradient immediately; a larger interval stores a running parameter-gradient sum.

Algorithm: NLT-prop forward-gradient updates.

Input: sequence $\boldsymbol{x}^{1:T}$, targets, Q, I, initial weights, update interval.
Initialize: $\boldsymbol{u}^0 = \boldsymbol{s}^0 = 0$; $\boldsymbol{\ell} = \boldsymbol{p} = 0$; $\boldsymbol{r}_\nu = \boldsymbol{q}_\nu = 0$.
Initialize: right direction $\boldsymbol{v} = \mathbf{1}/\sqrt{N}$ and gradient accumulator.
for $t = 1, \dots, T$ **do**
  Save old factors $\boldsymbol{\ell}^-, \boldsymbol{p}^-, \boldsymbol{r}_\nu^-, \boldsymbol{q}_\nu^-$.
  Set $\boldsymbol{\zeta}^t = [\boldsymbol{s}^{t-1}; \boldsymbol{x}^t]$ and compute $\boldsymbol{u}^t, \boldsymbol{s}^t$.
  Set $\boldsymbol{d}^t = \lambda\mathbf{1} + (\mathrm{diag}(\boldsymbol{W}) - \vartheta\mathbf{1}) \odot \boldsymbol{\psi}^{t-1}$.
  $(\boldsymbol{\ell}, \boldsymbol{p}) \leftarrow \mathrm{Pair}(\boldsymbol{d}^t \odot \boldsymbol{\ell}^-, \boldsymbol{p}^-, \mathbf{1}, \boldsymbol{\zeta}^t)$.
  $(\boldsymbol{\ell}, \boldsymbol{p}) \leftarrow \mathrm{Clip}(\boldsymbol{\ell}, \boldsymbol{p})$.
  if $Q > 0$ **then**
    $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{v}) \leftarrow \mathrm{Power}(\boldsymbol{C}^t, I, \boldsymbol{v})$.
    Define $\widetilde{\boldsymbol{C}}^t\boldsymbol{x} = \boldsymbol{a}(\boldsymbol{b}^\top\boldsymbol{x}) - \boldsymbol{a} \odot \boldsymbol{b} \odot \boldsymbol{x}$.
    for $\nu = 1, \dots, Q$ **do**
      if $\nu = 1$ **then**
        $(\boldsymbol{x}_2, \boldsymbol{y}_2, \cdot) \leftarrow \mathrm{Power}($
          $\widetilde{\boldsymbol{C}}^t\,\mathrm{Diag}(\boldsymbol{\ell}^-), I, \mathrm{normalize}(\boldsymbol{b} \odot \boldsymbol{\ell}^-))$.
      else $(\boldsymbol{x}_2, \boldsymbol{y}_2) \leftarrow (\widetilde{\boldsymbol{C}}^t\boldsymbol{r}_{\nu-1}^-, \boldsymbol{q}_{\nu-1}^-)$.
      end if
      $(\boldsymbol{r}_\nu, \boldsymbol{q}_\nu) \leftarrow \mathrm{Pair}(\boldsymbol{d}^t \odot \boldsymbol{r}_\nu^-, \boldsymbol{q}_\nu^-, \boldsymbol{x}_2, \boldsymbol{y}_2)$.
      $(\boldsymbol{r}_\nu, \boldsymbol{q}_\nu) \leftarrow \mathrm{Clip}(\boldsymbol{r}_\nu, \boldsymbol{q}_\nu)$.
    end for
  end if
  if a loss is supplied at t **then**
    Compute the current learning signal $\boldsymbol{g}^t$.
    Set $\boldsymbol{z}^t = \boldsymbol{g}^t \odot \boldsymbol{\ell} + \sum_{\nu=1}^{Q}((\boldsymbol{g}^t)^\top\boldsymbol{r}_\nu)\boldsymbol{q}_\nu$.
    Accumulate $\widehat{\nabla}_{\boldsymbol{\Theta}}\mathcal{L}^t = \boldsymbol{z}^t\boldsymbol{p}^\top$ and readout gradients.
  end if
  At an update boundary, apply the optimizer
  and clear the gradient accumulator.
end for

A.8 NLT-PROP FOR VGG-F

A.8.1 LAYER DYNAMICS AND THE FEEDBACK OPERATOR

Let $\boldsymbol{K}_l$ denote the linear convolution operator at layer l, and let $\boldsymbol{P}_l$ be the pooling operator before that convolution, or the identity when pooling is absent. The presynaptic maps are $\boldsymbol{x}_1^t = \boldsymbol{x}^t$, $\boldsymbol{x}_l^t = \boldsymbol{P}_l(\kappa\boldsymbol{s}_{l-1}^t)$ for $l > 1$, and $\boldsymbol{x}_F^t = \boldsymbol{R}(\kappa\boldsymbol{s}_L^{t-1})$ for feedback. The neuronal dynamics are

$$\begin{aligned}
\boldsymbol{u}_1^t &= \lambda_1\boldsymbol{u}_1^{t-1} + \boldsymbol{K}_1\boldsymbol{x}_1^t + \boldsymbol{F}\boldsymbol{x}_F^t - \vartheta_1\boldsymbol{s}_1^{t-1},\\
\boldsymbol{u}_l^t &= \lambda_l\boldsymbol{u}_l^{t-1} + \boldsymbol{K}_l\boldsymbol{x}_l^t - \vartheta_l\boldsymbol{s}_l^{t-1}, \qquad 2 \le l \le L,\\
\boldsymbol{s}_l^t &= H(\boldsymbol{u}_l^t - \vartheta_l\mathbf{1}).
\end{aligned} \tag{97}$$

Pooling is average pooling and feedback resizing uses nearest-neighbor interpolation. Their transpose operators respectively distribute a pooled value uniformly over its input window and sum the replicated entries of a resized map.

Define $\boldsymbol{\Psi}_l^t = \mathrm{Diag}(\boldsymbol{\psi}_l^t)$ and the current spatial transition $\boldsymbol{T}_l^t = \boldsymbol{K}_l\boldsymbol{P}_l\kappa\boldsymbol{\Psi}_{l-1}^t$ for $l > 1$. The spatial Jacobian from layer l to the final layer is

$$\boldsymbol{A}_l^t = \boldsymbol{T}_L^t\boldsymbol{T}_{L-1}^t\cdots\boldsymbol{T}_{l+1}^t, \qquad \boldsymbol{A}_L^t = \boldsymbol{I}. \tag{98}$$

To derive these products, perturb the previous states while fixing the current external input and weights. Let $\boldsymbol{B}_F^t = \boldsymbol{F}\boldsymbol{R}\kappa\boldsymbol{\Psi}_L^{t-1}$. The within-step perturbations satisfy

$$\delta\boldsymbol{u}_1^t = \boldsymbol{D}_1^t\delta\boldsymbol{u}_1^{t-1} + \boldsymbol{B}_F^t\delta\boldsymbol{u}_L^{t-1}, \tag{99}$$

$$\delta\boldsymbol{u}_2^t = \boldsymbol{D}_2^t\delta\boldsymbol{u}_2^{t-1} + \boldsymbol{T}_2^t\delta\boldsymbol{u}_1^t \tag{100}$$

$$= \boldsymbol{D}_2^t\delta\boldsymbol{u}_2^{t-1} + \boldsymbol{T}_2^t\boldsymbol{D}_1^t\delta\boldsymbol{u}_1^{t-1} + \boldsymbol{T}_2^t\boldsymbol{B}_F^t\delta\boldsymbol{u}_L^{t-1}. \tag{101}$$

Define $\boldsymbol{A}^t_{l,k} = \boldsymbol{T}^t_l \cdots \boldsymbol{T}^t_{k+1}$ for $k < l$ and $\boldsymbol{A}^t_{l,l} = \boldsymbol{I}$. Continuing the substitution through layer l gives

$$\delta \boldsymbol{u}^t_l = \sum_{k=1}^{l} \boldsymbol{A}^t_{l,k} \boldsymbol{D}^t_k \delta \boldsymbol{u}^{t-1}_k + \boldsymbol{A}^t_{l,1} \boldsymbol{B}^t_F \delta \boldsymbol{u}^{t-1}_L, \tag{102}$$

$$\boldsymbol{J}^t_{lk} = \begin{cases} \boldsymbol{A}^t_{l,k} \boldsymbol{D}^t_k, & k \le l,\ k < L, \\ \boldsymbol{A}^t_{l,1} \boldsymbol{B}^t_F, & k = L,\ l < L, \\ \boldsymbol{D}^t_L + \boldsymbol{A}^t_{L,1} \boldsymbol{B}^t_F, & k = l = L, \\ 0, & l < k < L, \end{cases} \tag{103}$$

$$\boldsymbol{J}^t_{LL} = \boldsymbol{D}^t_L + \boldsymbol{A}^t_{L,1} \boldsymbol{B}^t_F = \boldsymbol{D}^t_L + \boldsymbol{G}^t. \tag{104}$$

Since $\boldsymbol{A}^t_{L,1} = \boldsymbol{A}^t_1$, substituting $\boldsymbol{B}^t_F$ gives the loop operator in Eq. (107). Transposition reverses the operator order:

$$(\boldsymbol{G}^t)^\top = (\boldsymbol{A}^t_1 \boldsymbol{F} \boldsymbol{R} \kappa \boldsymbol{\Psi}^{t-1}_L)^\top \tag{105}$$

$$= \kappa \boldsymbol{\Psi}^{t-1}_L \boldsymbol{R}^\top \boldsymbol{F}^\top (\boldsymbol{A}^t_1)^\top. \tag{106}$$

The diagonal of $\boldsymbol{G}^t$ represents a full traversal from the final layer through feedback and back to that layer. It is retained in the nonlocal operator. This is why the VGG-F approximation uses $\boldsymbol{a}^t(\boldsymbol{b}^t)^\top$ without the RSNN off-diagonal mask.

The spatial operators use the current effective kernels and the surrogate derivatives at their indicated time indices. The complete feedback traversal from $\boldsymbol{u}^{t-1}_L$ to $\boldsymbol{u}^t_L$ is therefore

$$\boldsymbol{G}^t = \boldsymbol{A}^t_1 \boldsymbol{F} \boldsymbol{R} \kappa \boldsymbol{\Psi}^{t-1}_L, \qquad (\boldsymbol{G}^t)^\top = \kappa \boldsymbol{\Psi}^{t-1}_L \boldsymbol{R}^\top \boldsymbol{F}^\top (\boldsymbol{A}^t_1)^\top. \tag{107}$$

Both products are evaluated through convolutions, transpose convolutions, pooling, resizing, and elementwise multiplication. A fixed number of power iterations gives $\boldsymbol{a}^t, \boldsymbol{b}^t$. The source-layer emission factor is $\boldsymbol{b}^t_l = (\boldsymbol{A}^{t-1}_l)^\top \boldsymbol{b}^t$: a perturbation generated in layer l at the previous step first reaches the previous final-layer state and then enters the current feedback loop.

A.8.2 Factor Updates and Convolutional Gradients

The local diagonal is $\boldsymbol{D}^t_l = \mathrm{Diag}(\lambda_l \mathbf{1} - \vartheta_l \boldsymbol{\psi}^{t-1}_l)$. For each convolution, flatten the presynaptic and postsynaptic maps only for notation. The local factor update is

$$\boldsymbol{\ell}^t_l (\boldsymbol{p}^t_l)^\top = \mathcal{P}^{(K)}_1 \left[(\boldsymbol{d}^t_l \odot \boldsymbol{\ell}^{t-1}_l)(\boldsymbol{p}^{t-1}_l)^\top + \mathbf{1}(\boldsymbol{x}^t_l)^\top \right]. \tag{108}$$

For a local kernel trace, index an output position by $\mathbf{x}$ and the connected input position by $\mathrm{src}(\mathbf{x}, \Delta)$. Differentiating the local update gives

$$Z^t_{l,o,c,\Delta,\mathbf{x}} = d^t_{l,o,\mathbf{x}} Z^{t-1}_{l,o,c,\Delta,\mathbf{x}} + x^t_{l,c,\mathrm{src}(\mathbf{x},\Delta)} \tag{109}$$

$$\approx d^t_{l,o,\mathbf{x}} \ell^{t-1}_{l,o,\mathbf{x}} p^{t-1}_{l,c,\mathrm{src}(\mathbf{x},\Delta)} + x^t_{l,c,\mathrm{src}(\mathbf{x},\Delta)}. \tag{110}$$

Representing all presynaptic and postsynaptic positions in a virtual matrix makes the last line a sum of two outer products. Applying $\mathcal{P}^{(K)}_1$ gives Eq. (108); kernel connectivity is then imposed by the source-position map in the gradient correlation.

The nonlocal representation projects temporal histories onto the final-layer state space. For the source at layer l, the previous spatial propagation followed by the current loop gives

$$\boldsymbol{G}^t \boldsymbol{A}^{t-1}_l \approx \boldsymbol{a}^t (\boldsymbol{b}^t)^\top \boldsymbol{A}^{t-1}_l = \boldsymbol{a}^t \left((\boldsymbol{A}^{t-1}_l)^\top \boldsymbol{b}^t \right)^\top = \boldsymbol{a}^t (\boldsymbol{b}^t_l)^\top, \tag{111}$$

$$\boldsymbol{G}^t \boldsymbol{A}^{t-1}_l \,\mathrm{Diag}(\boldsymbol{\ell}^{t-1}_l) \approx \boldsymbol{a}^t (\boldsymbol{b}^t_l \odot \boldsymbol{\ell}^{t-1}_l)^\top, \tag{112}$$

$$\boldsymbol{G}^t \widehat{\boldsymbol{H}}^{t-1}_{l,\nu-1} \approx \boldsymbol{a}^t (\boldsymbol{b}^t)^\top \boldsymbol{r}^{t-1}_{l,\nu-1} (\boldsymbol{q}^{t-1}_{l,\nu-1})^\top \tag{113}$$

$$= \boldsymbol{a}^t \left(((\boldsymbol{b}^t)^\top \boldsymbol{r}^{t-1}_{l,\nu-1}) \boldsymbol{q}^{t-1}_{l,\nu-1} \right)^\top. \tag{114}$$

Propagating each stored left factor with $\boldsymbol{D}_L^t$ and adding the corresponding source therefore gives

$$\boldsymbol{H}_{l,\nu,\mathrm{cand}}^t = (\boldsymbol{d}_L^t \odot \boldsymbol{r}_{l,\nu}^{t-1})(\boldsymbol{q}_{l,\nu}^{t-1})^\top + \boldsymbol{a}^t(\boldsymbol{v}_{l,\nu}^t)^\top, \tag{115}$$

$$\widehat{\boldsymbol{H}}_{l,\nu}^t = \mathcal{P}_1^{(K)}[\boldsymbol{H}_{l,\nu,\mathrm{cand}}^t] = \boldsymbol{r}_{l,\nu}^t(\boldsymbol{q}_{l,\nu}^t)^\top. \tag{116}$$

Here $\boldsymbol{v}_{l,1}^t = \boldsymbol{b}_l^t \odot \boldsymbol{\ell}_l^{t-1}$, and $\boldsymbol{v}_{l,\nu}^t = ((\boldsymbol{b}^t)^\top \boldsymbol{r}_{l,\nu-1}^{t-1})\boldsymbol{q}_{l,\nu-1}^{t-1}$ for $\nu > 1$. The current spatial learning signal contracts the local history, and the final-layer learning signal contracts the stored nonlocal history:

$$z_{l,i}^t = g_{l,i}^t \ell_{l,i}^t + \sum_{\nu=1}^{Q} \sum_{k=1}^{n_L} g_{L,k}^t r_{l,\nu,k}^t q_{l,\nu,i}^t \tag{117}$$

$$= g_{l,i}^t \ell_{l,i}^t + \sum_{\nu=1}^{Q} ((\boldsymbol{g}_L^t)^\top \boldsymbol{r}_{l,\nu}^t) q_{l,\nu,i}^t. \tag{118}$$

NLT-prop represents nonlocal histories in the final-layer state space. Their stored left factors are propagated through $\boldsymbol{D}_L^t$, while additional loop traversals use $\boldsymbol{a}^t(\boldsymbol{b}^t)^\top$. At the first order, the source is $\boldsymbol{a}^t(\boldsymbol{b}_l^t \odot \boldsymbol{\ell}_l^{t-1})^\top$. At order $\nu > 1$, it is $\boldsymbol{a}^t((\boldsymbol{b}^t)^\top \boldsymbol{r}_{l,\nu-1}^{t-1})(\boldsymbol{q}_{l,\nu-1}^{t-1})^\top$. Compressing each source together with the propagated previous mode gives Eq. (7). This projection onto final-layer modes defines the VGG-F approximation of nonlocal history.

For an effective kernel entry connecting input channel c to output channel o at offset Δ, Eq. (8) reads

$$\widehat{\nabla}_{K_{l,o,c,\Delta}} \mathcal{L}^t = \sum_{\mathbf{x}} z_{l,o,\mathbf{x}}^t \, p_{l,c,\mathrm{src}(\mathbf{x},\Delta)}^t. \tag{119}$$

The source-position map src accounts for stride, padding, and dilation, with padded entries contributing zero. Gradients are averaged over samples in a batch. For the feedback kernel, use $\boldsymbol{x}_F^t$, the first-layer local signal, $\boldsymbol{b}_1^t$, and the final-layer nonlocal learning signal. Current spatial learning signals are computed within the present time step, including the classifier and any current activity penalty.

When kernels use scaled weight standardization, the trace calculation operates on the effective kernel. For one output channel with n incoming kernel entries, raw weights $\boldsymbol{w}$, centered weights $\boldsymbol{c} = \boldsymbol{w} - \overline{w}\mathbf{1}$, standard deviation $s = \sqrt{\|\boldsymbol{c}\|_2^2/n + \epsilon}$, standardized weights $\boldsymbol{z} = \boldsymbol{c}/s$, scale α, and fixed gain g_0, the effective kernel is $\widetilde{\boldsymbol{w}} = \alpha g_0 \boldsymbol{z}/\sqrt{n}$. An effective-kernel gradient $\boldsymbol{h}$ maps to

$$\widehat{\nabla}_{\boldsymbol{w}} \mathcal{L}^t = \frac{\alpha g_0}{\sqrt{n}\, s} \left(\boldsymbol{h} - \overline{h}\mathbf{1} - \boldsymbol{z}\,\overline{hz}\right), \qquad \widehat{\nabla}_{\alpha} \mathcal{L}^t = \frac{g_0}{\sqrt{n}} \boldsymbol{h}^\top \boldsymbol{z}, \tag{120}$$

The parameterization derivative is obtained entry by entry. For raw coordinate w_j,

$$\frac{\partial c_i}{\partial w_j} = \delta_{ij} - \frac{1}{n}, \tag{121}$$

$$\frac{\partial s}{\partial w_j} = \frac{1}{2s}\frac{2}{n}\sum_i c_i(\delta_{ij} - 1/n) = \frac{c_j}{ns}, \tag{122}$$

$$\frac{\partial z_i}{\partial w_j} = \frac{\delta_{ij} - 1/n}{s} - \frac{c_i}{s^2}\frac{c_j}{ns} = \frac{\delta_{ij} - 1/n - z_i z_j/n}{s}, \tag{123}$$

$$\frac{\partial \mathcal{L}^t}{\partial w_j} = \sum_i h_i \frac{\alpha g_0}{\sqrt{n}} \frac{\partial z_i}{\partial w_j} = \frac{\alpha g_0}{\sqrt{n}\, s} \left(h_j - \overline{h} - z_j \overline{hz}\right), \tag{124}$$

$$\frac{\partial \mathcal{L}^t}{\partial \alpha} = \sum_i h_i \frac{g_0 z_i}{\sqrt{n}} = \frac{g_0}{\sqrt{n}} \boldsymbol{h}^\top \boldsymbol{z}. \tag{125}$$

The bars denote means over the incoming kernel entries. This chain rule maps the online gradient to the parameters updated by the optimizer.

The VGG-F version of the pseudocode replaces the RSNN operator products with $\boldsymbol{G}^t$ and $(\boldsymbol{G}^t)^\top$, backprojects $\boldsymbol{b}^t$ through the previous spatial operators to obtain $\boldsymbol{b}_l^t$, and updates all convolutional and feedback traces from their saved factors. It then computes $\mathrm{Corr}(\boldsymbol{p}_l^t, \boldsymbol{z}_l^t)$, applies the scaled-weight-standardization chain rule, and submits those gradients to the same online optimizer. All convolutional layers share the single loop factorization at each time step.

A.8.3 Eligibility Storage

Let n_l be the number of entries in layer l and m_l the size of its presynaptic map. Each convolution stores $n_l + m_l$ local factors and $Q(n_L + n_l)$ nonlocal factors. Including the feedback trace, the total is

$$\sum_{l=1}^{L} \left[n_l + m_l + Q(n_L + n_l) \right] + n_1 + m_F + Q(n_L + n_1) \tag{126}$$

scalars per sample, where m_F is the resized feedback-input size. For fixed L and Q, this is linear in the feature-map sizes. Warm-start directions and temporary operator products add linear workspace. For the lateral model, the corresponding factor count is $(2Q + 2)N + M$.

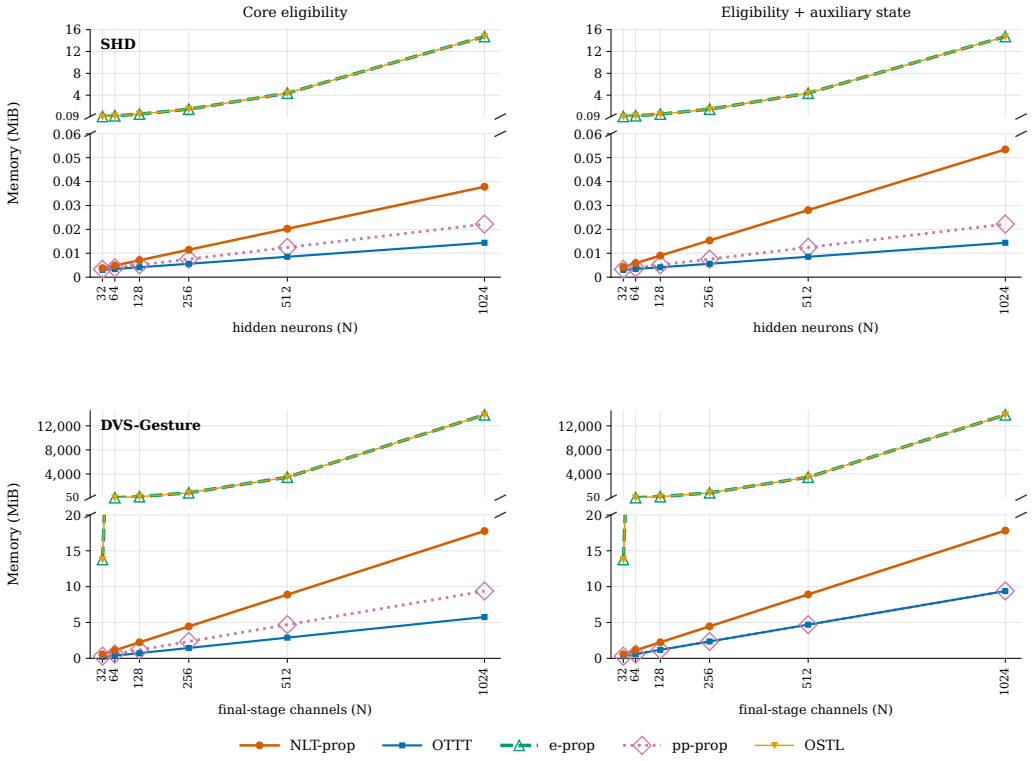


Figure 2: Eligibility-trace storage versus network width on SHD with RSNN (top) and DVS Gesture with VGG-F (sWS) (bottom). The left column counts core eligibility factors; the right also includes auxiliary trace state. Measurements use one input sequence with $B = 1$, $T = 100$, and FP32 storage. NLT-prop uses $Q = 1$ on SHD and $Q = 2$ on DVS Gesture. The horizontal axis denotes hidden neurons per layer for RSNN and final-stage channels for VGG-F, with channel ratios preserved. Each vertical axis has two linear segments separated by a break. The synapse-specific e-prop and OSTL curves overlap.

A.9 Choosing the Nonlocal Order

For a sequence of length T, the path expansion has orders $0 \le Q \le T - 1$. Increasing Q reduces the omitted temporal credit and introduces another compressed contribution. The useful range is governed by the balance between these two effects.

Proposition 1 (truncation and approximation error). Suppose $\|\boldsymbol{D}^t\|_2 \le d < 1$, $\|\boldsymbol{C}^t\|_2 \le c$, and $\|\boldsymbol{B}^t\|_F \le b$, with $\eta = c/(1-d) < 1$. Let $a = b/(1-d)$, and let δ_ν bound the approximation error

of the retained order ν. Then

$$\left\| \boldsymbol{E}^t - \sum_{\nu=0}^{Q} \widehat{\boldsymbol{E}}_\nu^t \right\|_F \leq \underbrace{\frac{a\eta^{Q+1}}{1-\eta}}_{\text{omitted paths}} + \underbrace{\sum_{\nu=0}^{Q} \delta_\nu}_{\text{retained-order errors}} . \tag{127}$$

The bound decreases when an added order satisfies $\delta_{Q+1} < a\eta^{Q+1}$. As the path envelope decays, the allowable approximation error also shrinks. The derivation below establishes this result, obtains δ_ν from the recursive approximation residuals, and gives a finite-horizon bound.

The corresponding gradient criterion is obtained directly. On a fixed trajectory, let $\boldsymbol{e}_Q$ be the difference between the full surrogate gradient and the order-Q estimate, and let $\boldsymbol{h}_{Q+1}$ be the newly added contribution. Adding that order reduces squared gradient error exactly when

$$2\langle \boldsymbol{e}_Q, \boldsymbol{h}_{Q+1} \rangle > \|\boldsymbol{h}_{Q+1}\|_2^2. \tag{128}$$

Thus an additional mode improves credit assignment when it recovers a sufficiently aligned part of the missing gradient. Successive Jacobian approximations and trace compressions affect this alignment. In the SHD ablation, mean accuracy over five seeds increases from 75.97% at $Q = 0$ to 76.83% at $Q = 1$, the highest mean among the tested orders 0 through 9. Higher orders produce nonmonotonic results, consistent with the interaction between recovered paths and accumulated approximation error.

A.9.1 Finite-Horizon Path Bound

Let $\|\boldsymbol{D}^s\|_2 \leq d$, $\|\boldsymbol{C}^s\|_2 \leq c$, and $\|\boldsymbol{B}^s\|_F \leq b$ along the trajectory. An injection at time τ traverses $h = t - \tau$ temporal transitions. For a binary vector $\boldsymbol{\omega} \in \{0,1\}^h$, set $\boldsymbol{A}_0^s = \boldsymbol{D}^s$, $\boldsymbol{A}_1^s = \boldsymbol{C}^s$, and $|\boldsymbol{\omega}| = \sum_j \omega_j$. Expanding every product gives

$$\boldsymbol{J}^t \cdots \boldsymbol{J}^{\tau+1} \boldsymbol{B}^\tau = \prod_{s=t}^{\tau+1} (\boldsymbol{D}^s + \boldsymbol{C}^s) \boldsymbol{B}^\tau \tag{129}$$

$$= \sum_{\boldsymbol{\omega} \in \{0,1\}^h} \boldsymbol{A}_{\omega_h}^t \cdots \boldsymbol{A}_{\omega_1}^{\tau+1} \boldsymbol{B}^\tau, \tag{130}$$

$$\boldsymbol{E}_\nu^t = \sum_{\tau=1}^{t-\nu} \sum_{|\boldsymbol{\omega}|=\nu} \boldsymbol{A}_{\omega_h}^t \cdots \boldsymbol{A}_{\omega_1}^{\tau+1} \boldsymbol{B}^\tau. \tag{131}$$

The product notation is ordered with decreasing time indices. The norm inequality $\|\boldsymbol{AB}\|_F \leq \|\boldsymbol{A}\|_2 \|\boldsymbol{B}\|_F$ applied successively yields

$$\|\boldsymbol{A}_{\omega_h}^t \cdots \boldsymbol{A}_{\omega_1}^{\tau+1} \boldsymbol{B}^\tau\|_F \leq \prod_{j=1}^{h} \|\boldsymbol{A}_{\omega_j}^{\tau+j}\|_2 \, \|\boldsymbol{B}^\tau\|_F \tag{132}$$

$$\leq b \prod_{j=1}^{h} d^{1-\omega_j} c^{\omega_j} = b d^{h-\nu} c^\nu. \tag{133}$$

There are $\binom{h}{\nu}$ choices of the ν nonlocal positions. Applying the triangle inequality and substituting $h = t - \tau$ gives

$$\|\boldsymbol{E}_\nu^t\|_F \leq \sum_{\tau=1}^{t-\nu} \binom{t-\tau}{\nu} b d^{t-\tau-\nu} c^\nu \tag{134}$$

$$= b \sum_{h=\nu}^{t-1} \binom{h}{\nu} d^{h-\nu} c^\nu. \tag{135}$$

Summing the omitted orders and changing the order of the finite sums gives

$$\left\| \sum_{\nu=Q+1}^{t-1} \boldsymbol{E}_\nu^t \right\|_F \le \sum_{\nu=Q+1}^{t-1} \|\boldsymbol{E}_\nu^t\|_F \tag{136}$$

$$\le b \sum_{\nu=Q+1}^{t-1} \sum_{h=\nu}^{t-1} \binom{h}{\nu} d^{h-\nu} c^\nu \tag{137}$$

$$= b \sum_{h=Q+1}^{t-1} \sum_{\nu=Q+1}^{h} \binom{h}{\nu} d^{h-\nu} c^\nu. \tag{138}$$

These finite sums apply for all nonnegative d, c, with an empty sum equal to zero.

A.9.2 Geometric Envelope and Proof of Proposition 1

Assume $0 \le d < 1$. Differentiating the absolutely convergent geometric series inside $|x| < 1$ gives

$$\sum_{h=0}^{\infty} x^h = \frac{1}{1-x}, \tag{139}$$

$$\frac{\mathrm{d}^\nu}{\mathrm{d}x^\nu} \sum_{h=0}^{\infty} x^h = \sum_{h=\nu}^{\infty} \frac{h!}{(h-\nu)!} x^{h-\nu}, \tag{140}$$

$$\frac{\mathrm{d}^\nu}{\mathrm{d}x^\nu} (1-x)^{-1} = \nu! (1-x)^{-\nu-1}, \tag{141}$$

$$\sum_{h=\nu}^{\infty} \binom{h}{\nu} x^{h-\nu} = (1-x)^{-\nu-1}. \tag{142}$$

Substitute $x = d$, extend the nonnegative finite sum to infinity, and define $a = b/(1-d)$ and $\eta = c/(1-d)$:

$$\|\boldsymbol{E}_\nu^t\|_F \le b c^\nu \sum_{h=\nu}^{\infty} \binom{h}{\nu} d^{h-\nu} = \frac{b c^\nu}{(1-d)^{\nu+1}} \tag{143}$$

$$= \frac{b}{1-d} \left(\frac{c}{1-d} \right)^\nu = a \eta^\nu. \tag{144}$$

When $\eta < 1$, the geometric tail satisfies

$$\left\| \sum_{\nu>Q} \boldsymbol{E}_\nu^t \right\|_F \le \sum_{\nu=Q+1}^{\infty} a \eta^\nu = a \eta^{Q+1} \sum_{j=0}^{\infty} \eta^j = \frac{a \eta^{Q+1}}{1-\eta}. \tag{145}$$

For $\|\boldsymbol{E}_\nu^t - \widehat{\boldsymbol{E}}_\nu^t\|_F \le \delta_\nu$, split the total error into omitted and retained orders:

$$\boldsymbol{E}^t - \sum_{\nu=0}^{Q} \widehat{\boldsymbol{E}}_\nu^t = \sum_{\nu>Q} \boldsymbol{E}_\nu^t + \sum_{\nu=0}^{Q} (\boldsymbol{E}_\nu^t - \widehat{\boldsymbol{E}}_\nu^t), \tag{146}$$

$$\left\| \boldsymbol{E}^t - \sum_{\nu=0}^{Q} \widehat{\boldsymbol{E}}_\nu^t \right\|_F \le \left\| \sum_{\nu>Q} \boldsymbol{E}_\nu^t \right\|_F + \sum_{\nu=0}^{Q} \|\boldsymbol{E}_\nu^t - \widehat{\boldsymbol{E}}_\nu^t\|_F \tag{147}$$

$$\le \frac{a \eta^{Q+1}}{1-\eta} + \sum_{\nu=0}^{Q} \delta_\nu =: \mathcal{B}_Q. \tag{148}$$

This proves Proposition 1. Keeping the same trajectory and orderwise error bounds, substitute $Q+1$ and subtract:

$$\mathcal{B}_{Q+1} - \mathcal{B}_Q = \left[\frac{a\eta^{Q+2}}{1-\eta} + \sum_{\nu=0}^{Q+1} \delta_\nu\right] - \left[\frac{a\eta^{Q+1}}{1-\eta} + \sum_{\nu=0}^{Q} \delta_\nu\right] \tag{149}$$

$$= \frac{a\eta^{Q+1}(\eta-1)}{1-\eta} + \delta_{Q+1} = \delta_{Q+1} - a\eta^{Q+1}. \tag{150}$$

The envelope decreases exactly when $\delta_{Q+1} < a\eta^{Q+1}$. The right side is the reduction in the omitted-path envelope, and the left side is the additional retained-order error budget.

A.9.3 Propagation of Approximation Residuals

Define $\boldsymbol{\Delta}_\nu^t = \widehat{\boldsymbol{E}}_\nu^t - \boldsymbol{E}_\nu^t$, with zero initial errors. The approximate recursion is the exact recursion plus its residual. Subtracting term by term gives

$$\boldsymbol{\Delta}_0^t = [\boldsymbol{D}^t \widehat{\boldsymbol{E}}_0^{t-1} + \boldsymbol{B}^t + \boldsymbol{R}_0^t] - [\boldsymbol{D}^t \boldsymbol{E}_0^{t-1} + \boldsymbol{B}^t] \tag{151}$$

$$= \boldsymbol{D}^t \boldsymbol{\Delta}_0^{t-1} + \boldsymbol{R}_0^t, \tag{152}$$

$$\boldsymbol{\Delta}_\nu^t = [\boldsymbol{D}^t \widehat{\boldsymbol{E}}_\nu^{t-1} + \boldsymbol{C}^t \widehat{\boldsymbol{E}}_{\nu-1}^{t-1} + \boldsymbol{R}_\nu^t] - [\boldsymbol{D}^t \boldsymbol{E}_\nu^{t-1} + \boldsymbol{C}^t \boldsymbol{E}_{\nu-1}^{t-1}] \tag{153}$$

$$= \boldsymbol{D}^t \boldsymbol{\Delta}_\nu^{t-1} + \boldsymbol{C}^t \boldsymbol{\Delta}_{\nu-1}^{t-1} + \boldsymbol{R}_\nu^t, \qquad \nu \geq 1. \tag{154}$$

Let $\|\boldsymbol{R}_\nu^t\|_F \leq \epsilon_\nu$. For the local order, repeated substitution yields

$$\|\boldsymbol{\Delta}_0^t\|_F \leq d\|\boldsymbol{\Delta}_0^{t-1}\|_F + \epsilon_0 \tag{155}$$

$$\leq d^2\|\boldsymbol{\Delta}_0^{t-2}\|_F + (1+d)\epsilon_0 \tag{156}$$

$$\leq \epsilon_0 \sum_{s=0}^{t-1} d^s \leq \frac{\epsilon_0}{1-d} =: \delta_0. \tag{157}$$

If $\|\boldsymbol{\Delta}_{\nu-1}^t\|_F \leq \delta_{\nu-1}$, then

$$\|\boldsymbol{\Delta}_\nu^t\|_F \leq d\|\boldsymbol{\Delta}_\nu^{t-1}\|_F + c\delta_{\nu-1} + \epsilon_\nu \tag{158}$$

$$\leq (c\delta_{\nu-1} + \epsilon_\nu) \sum_{s=0}^{t-1} d^s \leq \frac{c\delta_{\nu-1} + \epsilon_\nu}{1-d} =: \delta_\nu, \tag{159}$$

$$\delta_\nu = \eta\delta_{\nu-1} + \frac{\epsilon_\nu}{1-d} \tag{160}$$

$$= \eta^2\delta_{\nu-2} + \frac{\eta\epsilon_{\nu-1} + \epsilon_\nu}{1-d} \tag{161}$$

$$= \eta^\nu\delta_0 + \frac{1}{1-d}\sum_{j=1}^{\nu} \eta^{\nu-j}\epsilon_j \tag{162}$$

$$= \frac{1}{1-d}\sum_{j=0}^{\nu} \eta^{\nu-j}\epsilon_j. \tag{163}$$

The residual decomposition in Eq. (90) identifies the sources of ϵ_ν: the Jacobian approximation, the first-order source approximation, trace compression, shared presynaptic factors, and factor clipping.

For a uniform retained-order bound $\delta_\nu \leq \delta$, use

$$\mathcal{B}_Q = \frac{a\eta^{Q+1}}{1-\eta} + (Q+1)\delta. \tag{164}$$

For $0 < \eta < 1$ and $0 < \delta < a$, the beneficial-addition condition becomes

$$\mathcal{B}_{Q+1} < \mathcal{B}_Q \quad \Longleftrightarrow \quad \delta < a\eta^{Q+1} \tag{165}$$

$$\Longleftrightarrow \quad \delta/a < \eta^{Q+1} \tag{166}$$

$$\Longleftrightarrow \quad \log(\delta/a) < (Q+1)\log\eta \tag{167}$$

$$\Longleftrightarrow \quad Q+1 < \frac{\log(\delta/a)}{\log\eta}. \tag{168}$$

The last inequality reverses direction because $\log \eta < 0$. Together with $Q \leq T - 1$, it describes the useful additions to this envelope. For $\eta = 0$, the nonlocal operator is zero. If the contraction assumptions do not hold, the finite-horizon bound in Eq. (138) remains available.

A.9.4 Gradient Alignment and the Effect of an Added Order

Let $\widehat{\boldsymbol{E}}^t_{\leq Q} = \sum_{\nu=0}^{Q} \widehat{\boldsymbol{E}}^t_\nu$. The induced gradient error obeys

$$\|(\boldsymbol{E}^t)^\top \boldsymbol{g}^t - (\widehat{\boldsymbol{E}}^t_{\leq Q})^\top \boldsymbol{g}^t\|_2 = \|(\boldsymbol{E}^t - \widehat{\boldsymbol{E}}^t_{\leq Q})^\top \boldsymbol{g}^t\|_2 \tag{169}$$

$$\leq \|\boldsymbol{E}^t - \widehat{\boldsymbol{E}}^t_{\leq Q}\|_2 \|\boldsymbol{g}^t\|_2 \tag{170}$$

$$\leq \|\boldsymbol{E}^t - \widehat{\boldsymbol{E}}^t_{\leq Q}\|_F \|\boldsymbol{g}^t\|_2 \leq \mathcal{B}_Q \|\boldsymbol{g}^t\|_2. \tag{171}$$

Now compare nested estimates on the same trajectory with identical lower-order factors. Write $\boldsymbol{e}_Q = (\boldsymbol{E}^t - \widehat{\boldsymbol{E}}^t_{\leq Q})^\top \boldsymbol{g}^t$ and $\boldsymbol{h}_{Q+1} = (\widehat{\boldsymbol{E}}^t_{Q+1})^\top \boldsymbol{g}^t$. Then

$$\boldsymbol{e}_{Q+1} = (\boldsymbol{E}^t - \widehat{\boldsymbol{E}}^t_{\leq Q} - \widehat{\boldsymbol{E}}^t_{Q+1})^\top \boldsymbol{g}^t = \boldsymbol{e}_Q - \boldsymbol{h}_{Q+1}, \tag{172}$$

$$\|\boldsymbol{e}_{Q+1}\|_2^2 = (\boldsymbol{e}_Q - \boldsymbol{h}_{Q+1})^\top (\boldsymbol{e}_Q - \boldsymbol{h}_{Q+1}) \tag{173}$$

$$= \|\boldsymbol{e}_Q\|_2^2 - \boldsymbol{e}_Q^\top \boldsymbol{h}_{Q+1} - \boldsymbol{h}_{Q+1}^\top \boldsymbol{e}_Q + \|\boldsymbol{h}_{Q+1}\|_2^2 \tag{174}$$

$$= \|\boldsymbol{e}_Q\|_2^2 - 2\langle \boldsymbol{e}_Q, \boldsymbol{h}_{Q+1}\rangle + \|\boldsymbol{h}_{Q+1}\|_2^2, \tag{175}$$

$$\|\boldsymbol{e}_{Q+1}\|_2^2 - \|\boldsymbol{e}_Q\|_2^2 = \|\boldsymbol{h}_{Q+1}\|_2^2 - 2\langle \boldsymbol{e}_Q, \boldsymbol{h}_{Q+1}\rangle. \tag{176}$$

Therefore the squared gradient error decreases exactly under Eq. (128). For nonzero $\boldsymbol{e}_Q$ and $\boldsymbol{h}_{Q+1}$, substituting the cosine formula gives

$$2\langle \boldsymbol{e}_Q, \boldsymbol{h}_{Q+1}\rangle > \|\boldsymbol{h}_{Q+1}\|_2^2 \tag{177}$$

$$\iff \quad 2\|\boldsymbol{e}_Q\|_2 \|\boldsymbol{h}_{Q+1}\|_2 \cos\angle(\boldsymbol{e}_Q, \boldsymbol{h}_{Q+1}) > \|\boldsymbol{h}_{Q+1}\|_2^2 \tag{178}$$

$$\iff \quad \cos\angle(\boldsymbol{e}_Q, \boldsymbol{h}_{Q+1}) > \frac{\|\boldsymbol{h}_{Q+1}\|_2}{2\|\boldsymbol{e}_Q\|_2}. \tag{179}$$

An added mode must have sufficient alignment with the missing gradient relative to its magnitude. This gives the gradient-level explanation for choosing Q conditionally. The two accuracy sweeps in Section **??** show the corresponding nonmonotonic behavior: the highest mean occurs at $Q = 1$ for RSNN on SHD and at $Q = 7$ for VGG-F on DVS Gesture, while both sweeps include decreases at larger orders.

B Hyperparameters and Experimental

B.1 Backbones and NLT-prop Configuration

The RSNNs contain two LIF layers, each with 1,024 neurons for SHD, 512 for N-MNIST, and 256 for DMTS. Each layer has dense lateral recurrent connections, and successive layers communicate through spikes. The readout receives membrane potentials from the final layer. Recurrent weights are initialized with scale 0.15 for SHD and DMTS and 0.20 for N-MNIST. Input, inter-layer, and readout scales are 1.0. Each scale specifies the standard deviation before division by the square root of the number of incoming connections. Recurrent diagonals are initialized to zero and remain trainable.

VGG-F follows the VGG-style convolutional backbone (Simonyan & Zisserman, 2015; Xiao et al., 2022) and uses the channel sequence $64, 128, \mathrm{P}, 256, 256, \mathrm{P}, 512, 512, \mathrm{P}, 512, 512$, where P denotes 2×2 average pooling. All feedforward convolutions use 3×3 kernels. A 3×3 convolution maps the resized final-layer spikes from the preceding time step to the first layer. Feedforward and readout initialization scales are 1.0, and the feedback initialization scale is 0.1. Scaled weight standardization (Brock et al., 2021; Xiao et al., 2022) uses gain 2.74 and numerical constant 10^{-5}, with learnable scales initialized to 1.0 for feedforward layers and 0.1 for feedback. The spike scale is 1.0. All six learning methods use this topology on the two VGG-F datasets.

Table 4: NLT-prop training hyperparameters. Start indices are zero-based. The update interval is measured in time steps.

Setting	SHD	N-MNIST	CIFAR10-DVS	DVS Gesture	DMTS
Leak factor λ	0.98	0.97	0.5	0.6	0.996
Threshold ϑ	1.0	1.0	0.75	1.0	1.0
Order Q	1	1	1	2	1
Jacobian power iterations	2	2	2	1	2
Initial learning rate	10^{-4}	5×10^{-5}	10^{-4}	3×10^{-4}	5×10^{-5}
Minimum learning rate	10^{-6}	10^{-6}	10^{-6}	3×10^{-6}	10^{-6}
Weight decay	3×10^{-4}	10^{-4}	10^{-4}	1.5×10^{-4}	10^{-4}
Training batch size	128	192	128	128	256
Evaluation batch size	256	384	256	256	512
Maximum epochs	80	100	180	100	30
Early-stopping patience	18	12	40	15	7
Label smoothing	0.02	0.02	0.05	0.02	0
Activity penalty strength	0.1	0.1	0.001	0.0005	0.001
Target firing rate	0.05	0.03	0.06	0.06	0.05
Loss start index	50	15	4	10	530
Readout start index	10	3	1	10	530
Update interval	5	3	2	1	1

All networks start from zero membrane potentials and use the triangular surrogate in Eq. (2), with width 1.0 and height 0.3. NLT-prop includes the reset derivative, clips stored trace factors at magnitude 20, and uses numerical constant 10^{-8} in its factorization operations. Table 4 lists its dataset-specific settings. Dense RSNN local factors use the two-dimensional compression described in Appendix A. VGG-F uses two compression iterations for CIFAR10-DVS and one for DVS Gesture.

Training uses AdamW (Loshchilov & Hutter, 2019) with moment coefficients $(0.9, 0.999)$ and a single cosine-annealing cycle (Loshchilov & Hutter, 2017) spanning the maximum number of epochs. We apply gradient norm clipping (Pascanu et al., 2013) with a threshold of 1.0. Weight magnitudes are bounded by 3.0 after each update. At supervised time steps, the loss combines cross-entropy with label smoothing (Szegedy et al., 2016) and the firing-rate penalty, using the coefficients in Table 4. Checkpoints are evaluated after each epoch, and early stopping uses an accuracy improvement threshold of 10^{-4}. The selected checkpoint is restored for the final evaluation.

B.2 Evaluation Metrics

For each example, logits are averaged from the configured readout start index to the end of the sequence. Applying softmax to these averaged logits yields the class probabilities, and their maximum determines the predicted class. Accuracy is the fraction of correctly classified examples.

B.3 Power-Iteration Ablation

We vary power iterations while holding $Q = 1$ for RSNN. The factorization rank remains one in both backbones. In RSNN, this parameter controls the iterations used to estimate the dominant factors of the off-diagonal Jacobian and the first-order nonlocal source. This ablation examines the computational effort devoted to estimating the factors at a fixed representation size.

On SHD, one, two, three, and four power iterations yield accuracies of $77.09 \pm 0.77\%$, $76.83 \pm 1.08\%$, $76.20 \pm 1.15\%$, and $76.86 \pm 2.12\%$, respectively, over five seeds. One iteration gives the highest mean among these settings. At the selected checkpoints, the mean cosine similarity between the hidden-parameter gradient and BPTT ranges from 0.553 to 0.571, with both gradients evaluated at identical weights, inputs, and losses. The SHD results favor a small iteration count: the extra matrix–vector products do not produce a consistent accuracy gain.

Table 5: Power-iteration ablation on SHD with RSNN at $Q = 1$. Accuracy is reported as mean $\pm$ sample standard deviation in percent over five seeds. The highest mean is bold, and the second-highest is underlined.

Power iterations I	**ACC (%, Mean$\pm$SD)**
1	**77.09$\pm$0.77**
2	76.83$\pm$1.08
3	76.20$\pm$1.15
4	<u>76.86$\pm$2.12</u>

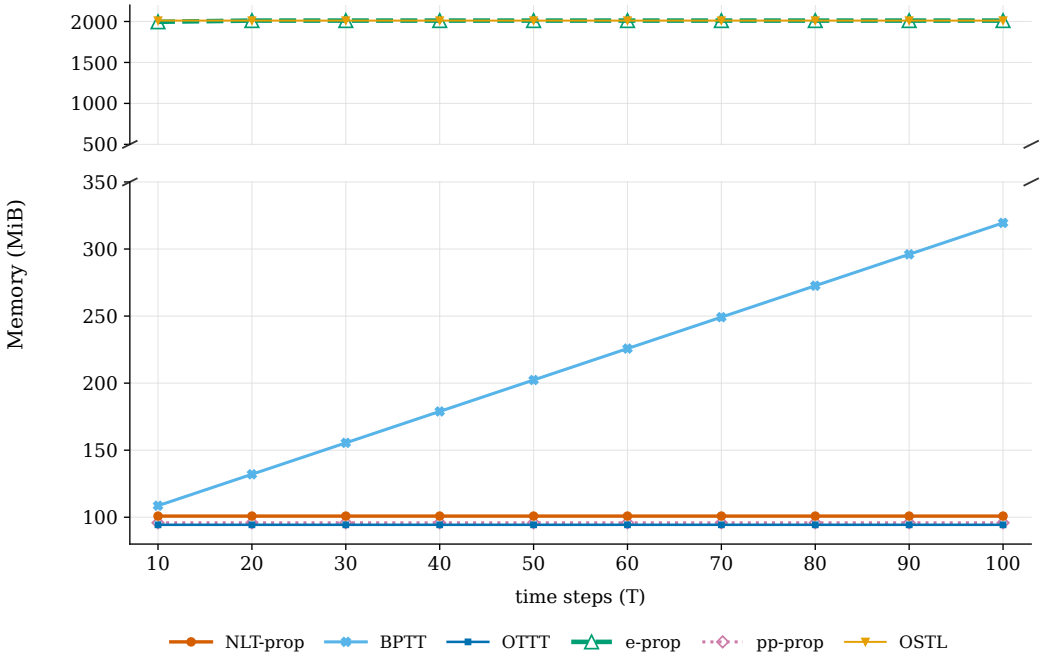


Figure 3: Peak allocated training GPU memory versus sequence length on SHD with RSNN. Each point is one complete training run with seed 42, batch size 128, and two layers of 1,024 neurons. The vertical axis uses two linear segments separated by an explicit break.

B.4 GPU Memory Across Sequence Lengths

Figure 3 reports peak allocated GPU memory during complete SHD training runs on an RTX 5090. We use the two-layer RSNN network with 1,024 neurons per layer, batch size 128, seed 42, and $T \in \{10, 20, \ldots, 100\}$. Each value of T defines the number of event bins per sample. Inputs remain on the CPU until their current frame is needed, and the measurement includes the model, gradients, optimizer state, temporal states, and operator workspace. NLT-prop uses $Q = 1$ and two Jacobian power iterations. All methods retain their configured learning rules and training schedules.

BPTT memory increases from 108.60 MiB at $T = 10$ to 319.48 MiB at $T = 100$. NLT-prop remains at 100.86 MiB throughout the sweep. The online baselines also have constant peaks over these sequence lengths: 94.36 MiB for OTTT and 1996.23-2,011.05 MiB for e-prop, 95.86 MiB for pp-prop, and 2,011.05 MiB for OSTL. This separates two memory effects: storing the temporal computation graph introduces growth with T, while the representation of eligibility traces determines the persistent storage required by online updates.

The e-prop implementation used in this figure stores the shared presynaptic LIF trace once, exploiting the identical postsynaptic rows of its trace recurrence. The dense synapse-specific representation

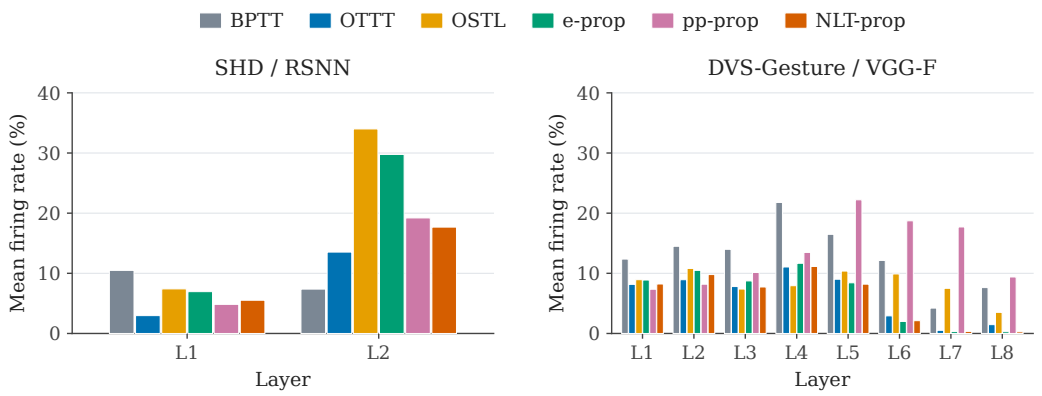


Figure 4: Mean firing rates for each spiking layer on SHD with RSNN and DVS Gesture with VGG-F. Bars show the mean over five seeds. Within each run, rates are the percentage of neuron–time pairs containing a spike, averaged over the complete test set.

and this shared representation are derived separately in Appendix A.8.3. OSTL retains a trace for each connection because its reset-dependent decay varies across postsynaptic neurons.

B.5 Layerwise Mean Firing Rates

For layer l with n_l neurons, define the spike fraction in run r as

$$f_l^{(r)} = \frac{100}{STn_l} \sum_{a=1}^{S} \sum_{t=1}^{T} \sum_{i=1}^{n_l} s_{l,i}^{t,(a,r)}, \qquad \overline{f}_l = \frac{1}{5} \sum_{r=1}^{5} f_l^{(r)}, \tag{180}$$

where S is the number of test examples. For convolutional layers, n_l includes channels and spatial positions. Figure 4 shows these percentages for the checkpoints used in the main comparison, averaging all 100 time steps and all test examples before averaging over seeds. Pooling and the membrane readout are excluded from the spiking-layer count.

On SHD, NLT-prop has mean firing rates of 5.52% and 17.68% in the two RSNN layers. Its second-layer rate is below those of e-prop (29.76%) and OSTL (34.00%), alongside its higher accuracy in Table 1. On DVS Gesture, the NLT-prop rates in the final three VGG-F layers are 2.13%, 0.34%, and 0.27%. The layerwise measurements show that NLT-prop maintains sparse activity in the deeper convolutional layers while retaining the competitive accuracy reported in the main comparison.

B.6 Gradient Alignment with BPTT

We compare the parameter gradients of the last hidden layer on SHD with RSNN and on DVS Gesture with VGG-F (sWS). After each of the first ten epochs of an NLT-prop training run with seed 42, all learning rules are evaluated at the same weights, using identical inputs, neuronal dynamics, and instantaneous losses. BPTT differentiates the current loss through the complete preceding trajectory. For method m, we measure the cosine similarity

$$\rho_m = \frac{\langle \boldsymbol{h}_m, \boldsymbol{h}_{\mathrm{BPTT}} \rangle}{\|\boldsymbol{h}_m\|_2 \, \|\boldsymbol{h}_{\mathrm{BPTT}}\|_2}, \tag{181}$$

where $\boldsymbol{h}_m$ and $\boldsymbol{h}_{\mathrm{BPTT}}$ are the flattened parameter gradients of layer 2 for RSNN and layer 8 for VGG-F, measured before optimizer gradient clipping. The VGG-F gradient includes the convolutional weights and learnable sWS scales. Each point in Figure 5 averages nine comparisons from three fixed minibatches of 16 examples, evaluated at zero-based time indices 50, 75, and 95. NLT-prop uses $(Q, I) = (1, 2)$ on SHD and $(Q, I) = (2, 1)$ on DVS Gesture, where I denotes the number of Jacobian power iterations.

On SHD, NLT-prop achieves the highest gradient cosine similarity at every measured epoch. Averaging the ten epoch values gives 0.759 for NLT-prop, compared with 0.729 for OSTL, 0.681 for pp-prop, and 0.381 for both OTTT and e-prop. At epoch 10, the corresponding values are 0.758, 0.726,

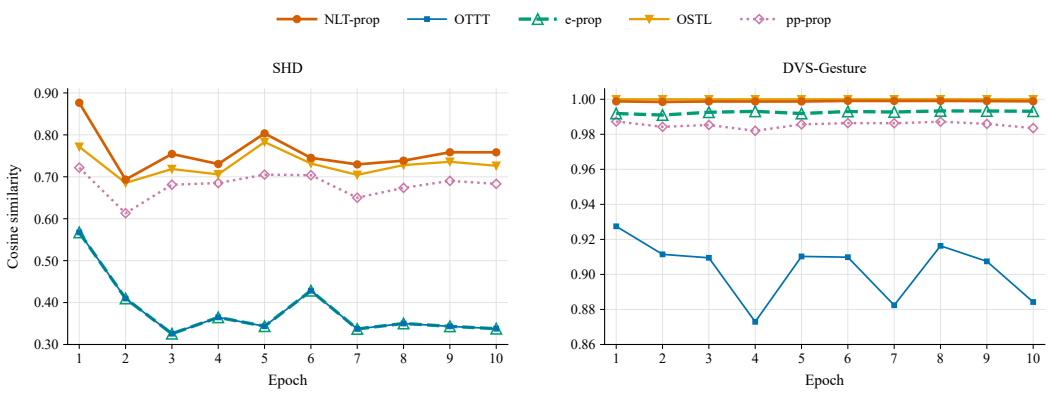


Figure 5: Last hidden layer parameter gradient cosine similarity with BPTT on SHD using RSNN (left) and DVS Gesture using VGG-F (sWS) (right). All methods share the weights from an NLT-prop training trajectory with seed 42. Each point is the mean of comparisons at the end of the indicated epoch.

0.683, and 0.338. Thus, NLT-prop retains stronger alignment with the BPTT gradient throughout the observed RSNN training trajectory.

On DVS Gesture, NLT-prop maintains a cosine similarity between 0.9985 and 0.9991, with a ten-epoch mean of 0.9988. OSTL is closest to BPTT, with a mean of 0.99998, followed by NLT-prop, e-prop at 0.9926, pp-prop at 0.9854, and OTTT at 0.9032. Across both backbones, NLT-prop consistently exceeds OTTT and pp-prop in last-layer gradient alignment. These results support the ability of its compact nonlocal representation to retain temporal information relevant to the BPTT gradient direction while maintaining linear eligibility-trace storage.