GRADIENT-AWARE LEARNABLE TEMPORAL TRUNCATION FOR BACK-PROPAGATION THROUGH TIME IN SPIKING NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Spiking Neural Networks (SNNs) have demonstrated promising data-processing capabilities with low energy consumption. Back-Propagation Through Time (BPTT) enables training based on Spatio-Temporal Back-Propagation (STBP) by propagating errors through temporal dynamics. Conventional BPTT aggregates gradients calculated at each time step and updates network parameters using the aggregated gradients, even though the status of SNNs varies across time steps. This shortcoming causes two problems: temporal mismatch in gradient updates and high memory usage during the backward pass. Conventional studies have suggested Truncated BPTT (T-BPTT) to overcome these problems. However, they employ a predetermined truncation strategy that fails to account for temporal variations in gradient behavior, resulting in poor learning performance. To address these problems, we investigate how temporal truncation affects the gradients used to train SNNs and find that effective truncation configurations consistently exhibit smaller gradient dispersion. Guided by this finding, we propose Learnable Truncated BPTT (LT-BPTT), which jointly adjusts the number and lengths of truncated temporal segments. LT-BPTT generates candidate temporal partitions, evaluates their gradient dispersion using a probe mini-batch, and selects the partition with the lowest gradient dispersion. Extensive experiments across diverse SNN architectures and datasets demonstrate that LT-BPTT achieves the highest classification accuracy in most evaluated settings while reducing memory consumption compared with BPTT and benchmarks to improve learning performance of SNNs.

1 INTRODUCTION

Spiking Neural Networks (SNNs) are third-generation neural networks using spikes to process data (Maass, 1997; Tan et al., 2020). However, their data-processing capability is inferior to that of Deep Neural Networks (DNNs) due to limitations in training methods, which prevent SNNs from learning data samples effectively (Deng et al., 2022b; Zhang et al., 2025; Yao et al., 2025). These limitations are significant drawbacks of SNNs, contributing to their relatively limited adoption compared with DNNs across various application domains. Nevertheless, SNNs are important technologies to allow neural networks to be deployed on portable devices, as these devices demand personalized Artificial Intelligence (AI) services (Moriya et al., 2025; Yin et al., 2026).

Addressing the limitations and extending SNN adoption to broader domains, conventional studies focus on applying supervised learning with back-propagation to SNNs and on developing Spatio-Temporal Back-Propagation (STBP). Indeed, STBP notably improves the learning performance of SNNs based on Back-Propagation Through Time (BPTT). **However, this learning approach faces two significant problems: the lack of a principled mechanism to determine the truncation length for network parameter updates and the high memory consumption of a single backward pass, which results from storing intermediate states across all time steps (Aicher et al., 2020; Guo et al., 2023).** In existing BPTT schemes for SNNs, parameter updates are performed only at the last time step, even though appropriate gradients for parameter updates are not guaranteed at that point (Guo et al., 2023). Furthermore, accumulating gradients across all time steps until the final step incurs substantial memory overhead during training. These problems severely degrade the learning performance and efficiency (Aicher et al., 2020).

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.

Researchers have developed Truncated BPTT (T-BPTT) to overcome these shortcomings. Indeed, T-BPTT reduces memory occupation compared with conventional BPTT while alleviating vanishing and exploding gradient problems (Aicher et al., 2020). **However, existing T-BPTT relies on a fixed, predefined truncation scheme that cannot adapt to varying temporal characteristics of gradients (Guo et al., 2023).** Fixed-length T-BPTT assumes that the same gradient propagation range is suitable throughout training. In practice, however, the required temporal learning range can vary across network architectures and training stages (Kang et al., 2026). If the truncation length is too short, important learning information from later time steps is missed. If it is too long, unnecessary or unstable gradients can accumulate. Therefore, using a single fixed truncation length can lead to unbalanced gradient propagation over time, resulting in suboptimal convergence.

To solve these problems, we propose a Learnable Truncated BPTT (LT-BPTT), which adaptively partitions the full sequence of time steps into shorter segments, truncates backward temporal dependencies at the segment boundaries, and accumulates the partition gradients before performing a single optimizer update for each mini-batch. We first investigate how including inappropriate time steps in gradient computation affects learning to determine appropriate time steps for truncation. **Our analysis reveals that gradients computed over such time steps exhibit greater dispersion, thereby degrading the SNNs' learning performance.** This is because large gradient dispersion indicates increasingly heterogeneous gradients, leading to an imbalanced allocation of the informative contribution to the gradient calculations. Based on our analysis, we design an adaptive mechanism that adjusts the number and lengths of truncated temporal segments to prevent excessive increases in gradient dispersion.

The objective of LT-BPTT is to minimize the dispersion of gradients calculated at each time step, thereby removing the imbalanced allocation of temporal contributions. To do so, LT-BPTT adjusts the temporal segments with the following two steps. First, it generates candidate partitions by splitting, merging, or shifting the boundaries of the current partition. Second, it evaluates candidate partitions using a probe mini-batch and selects the partition that yields the lowest gradient dispersion. Third, it repeats the evaluation and selection during training. The selected partition is then used for TBPTT during the subsequent training interval. Through these steps, LT-BPTT jointly adapts both the number of temporal segments and their lengths.

With LT-BPTT, SNNs improve their learning performance without complex methods. Our empirical observations support that SNNs using LT-BPTT achieve superior learning performance compared to SNNs using similar benchmarks. We conduct extensive experiments across diverse SNN architectures and datasets, comparing LT-BPTT with various benchmarks. In most experimental settings, SNNs equipped with LT-BPTT achieve the highest classification accuracy while consuming less memory. Our results suggest that LT-BPTT simultaneously enhances the learning performance and efficiency of SNNs. Our work has the following contributions.

- Unlike conventional BPTT, we propose an adaptive temporal partitioning framework that jointly determines the number of temporal segments and the length of each segment to ensure appropriate gradient updates.
- Through a motivation study, we provide empirical evidence and theoretical analysis showing that inappropriate temporal partitions increase gradient dispersion and consequently degrade the learning performance of SNNs; guided by this finding, we design a mechanism that adaptively selects temporal partitions by minimizing gradient standard deviation.
- By avoiding the computation chain over the entire time horizon as in conventional BPTT, LT-BPTT reduces the memory required to store temporal activations and intermediate states while improving the learning performance of SNNs.

2 BACKGROUNDS

2.1 SPIKING NEURAL NETWORKS

SNNs are neural networks in which units communicate using discrete spikes and maintain internal temporal states. They are often categorized as third-generation neural networks because the timing of spikes is part of the computation, not merely the magnitude of a static activation (Maass, 1997). A commonly used neuron model in SNNs is the Leaky Integrate-and-Fire (LIF) neuron, in which

synaptic input accumulates in the membrane potential, the potential decays over time, and a binary spike is emitted once the threshold is crossed (Gerstner & Kistler, 2002). This dynamics makes an SNN equivalent to a recurrent computational graph even when its spatial connectivity is feed-forward. Information can be encoded through firing rates, spike latencies, precise spike times, or population activity, allowing SNNs to process both static inputs converted into spike trains and naturally event-based streams. Since spike events are sparse and asynchronous, SNNs are well matched to neuromorphic hardware and event-based sensors, but their non-differentiable spike generation also complicates gradient-based optimization (Neftci et al., 2019; Eshraghian et al., 2023).

2.2 Back-Propagation Through Time

BPTT is the standard learning method for recurrent models. In Recurrent Neural Networks (RNNs) and SNNs, losses are attached to one or more time steps, and ordinary backpropagation is applied to the expanded computational graph (Werbos, 1990). In SNNs, BPTT differentiates through the sequence of membrane potentials, synaptic currents, and spikes, so each parameter receives gradient contributions from both current-layer interactions and past neuronal states (Wu et al., 2018). BPTT is conceptually simple but requires substantial resources because intermediate states must be stored until the backward pass, and the backward graph grows with the simulation length. It can also suffer from vanishing or exploding gradients (Hochreiter & Schmidhuber, 1997).

2.2.1 Truncated Back-Propagation Through Time

Truncated BPTT reduces memory and computation by detaching the graph after a fixed temporal window, making training more practical for long sequences or online settings in RNNs (Williams & Peng, 1990).

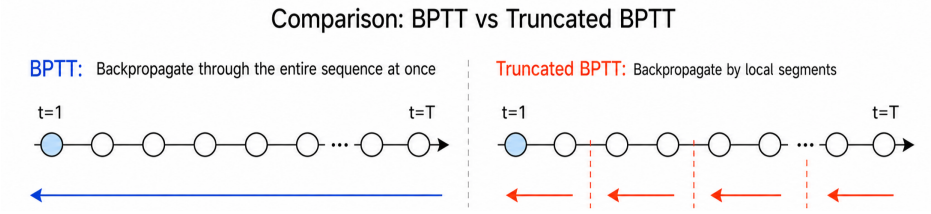

Figure 1: Comparison between normal BPTT and truncated BPTT.

Figure 1 shows the difference between normal BPTT and truncated BPTT. Different from normal BPTT, truncated BPTT uses temporal segments to compute gradients and update network parameters. However, truncating BPTT with fixed segment lengths biases the gradient toward short-term dependencies and can discard useful long-range credit assignment (Tallec & Ollivier, 2017). Adaptive truncation addresses this limitation by selecting the backward horizon through estimated gradient bias or temporal contribution, rather than fixing the same window across all training phases and samples (Aicher et al., 2020).

3 State of the Art

3.1 Methods to train deep SNNs effectively

Deep SNNs are mainly trained through surrogate-gradient learning, ANN-to-SNN conversion, and architectural normalization methods (Zhou et al., 2024). Surrogate gradients bypass the non-differentiable spike function and enable spatio-temporal information assignment in multilayer SNNs (Zenke & Ganguli, 2018; Wu et al., 2018; Neftci et al., 2019). Adaptive or learnable surrogates refine the backward proxy instead of fixing the back-propagation method (Li et al., 2021b; Wang et al., 2023; Lian et al., 2023). Normalization methods such as threshold-dependent Batch Normalization (tdBN) and Temporal Effective Batch Normalization (TEBN) stabilize direct training across depth and time (Zheng et al., 2021; Duan et al., 2022). Spike-compatible residual designs such as SEW-ResNet support much deeper directly trained SNNs (Fang et al., 2021). Conversion methods improve low-latency inference by calibrating ANN-derived thresholds and membrane dynamics (Li et al., 2021a; Bu et al., 2022; Hao et al., 2023). Temporal objectives such as TET further exploit multi-step outputs during training (Deng et al., 2022b).

3.2 EFFORTS TO IMPROVE BPTT

BPTT optimizes temporal dynamics by unrolling them, but its training cost grows with the number of time steps (Werbos, 1990; Wu et al., 2018; Xiao et al., 2022b). T-BPTT reduces this cost by back-propagating only through a recent temporal window (Williams & Peng, 1990; Tallec & Ollivier, 2017). However, a fixed window introduces truncation bias and weakens long-range credit assignment (Tallec & Ollivier, 2017; Aicher et al., 2020). Anticipated Reweighted Truncated Back-Propagation (ARTBP) mitigates this bias using random truncation and compensation factors (Tallec & Ollivier, 2017). Adaptive TBPTT (A-TBPTT) selects the truncation length from a target gradient-bias in Recurrent Neural Networks (RNNs) (Aicher et al., 2020). For SNNs, temporally truncated local BPTT combines temporal truncation with local losses to reduce training cost (Guo et al., 2023). Online or selective methods such as e-prop, Online Training Through Time (OTTT), and Spatial Learning Through Time (SLTT) reduce reliance on full backward-in-time traversal (Bellec et al., 2020; Xiao et al., 2022b; Meng et al., 2023a).

4 MOTIVATION STUDY

We study how gradients change in SNNs with various BPTT truncation configurations. We empirically and analytically investigate how the TBPTT truncation length alters gradients in SNNs. Across the evaluated model-dataset pairs, the truncation length that achieves the highest accuracy consistently has the smallest average coordinate-wise gradient standard deviation.

4.1 EMPIRICAL PREMISE ABOUT THE TRUNCATION LENGTH

Across different truncation lengths, we observe that the best-performing truncation length produces the smallest gradient standard deviation. This strong empirical correspondence suggests that controlling gradient dispersion may be an important factor in determining an effective truncation length for TBPTT. For statistical reliability, we repeat the experiments 10 times and present the results with 95% confidence intervals.

Table 1: Classification accuracy (%) and average standard deviation of gradients over time steps for various SNN models. We set the number of time steps to 8 and truncation length k to 1, 2, 4, and 8.

	K=1	K=2	K=4	K=8 (FULL BPTT)
MLP WITH F-MNIST				
ACCURACY	87.8 ± 0.96	88.2 ± 0.81	88.88 ± 0.74	$\mathbf{89.22 \pm 0.57}$
STANDARD DEVIATION	$(3.57 \pm 0.27) \times 10^{-4}$	$(2.37 \pm 0.22) \times 10^{-4}$	$(1.78 \pm 0.21) \times 10^{-4}$	$(1.08 \pm 0.19) \times 10^{-4}$
VGG-11 WITH CIFAR-10				
ACCURACY	85.64 ± 2.33	$\mathbf{87.53 \pm 1.98}$	83.94 ± 2.54	81.19 ± 2.79
STANDARD DEVIATION	$(6.14 \pm 0.61) \times 10^{-5}$	$\mathbf{(5.46 \pm 0.57) \times 10^{-5}}$	$(3.69 \pm 0.28) \times 10^{-4}$	$(2.06 \pm 0.25) \times 10^{-2}$
RESNET-18 WITH CIFAR-100				
ACCURACY	63.38 ± 4.98	70.72 ± 3.95	$\mathbf{71.86 \pm 3.73}$	70.75 ± 3.66
STANDARD DEVIATION	$(3.61 \pm 0.31) \times 10^{-4}$	$(1.66 \pm 0.26) \times 10^{-4}$	$\mathbf{(9.77 \pm 0.49) \times 10^{-5}}$	$(1.05 \pm 0.38) \times 10^{-4}$

Table 1 presents the classification accuracy and standard deviation of various SNN models and truncation lengths. The classification accuracy increases as the standard deviation of all gradients in the SNN models decreases. This experimental result suggests that selecting a truncation length k that reduces the gradient standard deviation is important for achieving strong learning performance. Compared to RNNs, spike-triggered reset and surrogate-gradient gating make temporal credit assignment highly uneven across neurons and time steps, causing T-BPTT performance to depend strongly on per-time-step gradient dispersion.

4.2 WHY GRADIENT DISPERSION DESTABILIZES TEMPORAL CREDIT AGGREGATION

As gradient dispersion increases, a small subset of temporal gradients grows disproportionately, causing different time steps to exhibit distinct dominant gradient components. This temporal inconsistency induces imbalanced credit assignment, where gradients from different time steps may cancel out or become excessively amplified during aggregation. Consequently, the resulting parameter updates become irregular and cannot consistently reduce the loss, degrading learning performance.

For a fixed truncation length k, let $\mathbf{g}_t^{(k)} = [g_{t,1}^{(k)}, \ldots, g_{t,d}^{(k)}]^\top \in \mathbb{R}^d$ denote the scalar gradients at timestep t, where all gradient tensors are flattened into a single vector. d denotes the total number of trainable scalar parameters in the network. The mean (μ_t), mean-centered gradient (h_t), and standard deviation (σ_t) of the gradient entries at timestep t are defined as

$$\mu_t = \frac{1}{d}\sum_{i=1}^{d} g_{t,i}, \qquad h_{t,i} = g_{t,i} - \mu_t, \qquad \mathbf{h}_t = [h_{t,1}, \ldots, h_{t,d}]^\top, \qquad \sigma_t^2 = \frac{1}{d}\sum_{i=1}^{d} h_{t,i}^2 = \frac{1}{d}\left\|\mathbf{h}_t\right\|_2^2 \tag{1}$$

where i indexes the gradients after vectorization. By construction, the single scalar value of the mean-centered gradient satisfies $\sum_{i=1}^{d} h_{t,i} = 0$.

The normalized aggregation of timestep-level gradients is defined as

$$\mathbf{G} = \sum_{t\in\mathcal{T}} w_t \mathbf{g}_t, \qquad \sum_{t\in\mathcal{T}} w_t = 1, \qquad w_t \geq 0. \tag{2}$$

Here, $\mathcal{T}$ is the set of time steps contributing to the gradient update, and w_t is the normalized weight assigned to the gradient at timestep t. Without normalization, a partition containing more timestep gradients can produce a larger update simply because there are different numbers of partitions in candidates and more gradients are summed. We normalize the weights so that different partitions can be compared independently of the number of included gradients.

To characterize whether the mean-centered gradient magnitude is broadly distributed over the single scalar values in gradient vectors, we introduce its effective support size:

$$a_t = \frac{1}{d}\left\|\mathbf{h}_t\right\|_1, \quad s_t^{\text{eff}} = \frac{\left\|\mathbf{h}_t\right\|_1^2}{\left\|\mathbf{h}_t\right\|_2^2} = d\left(\frac{a_t}{\sigma_t}\right)^2. \tag{3}$$

Here, a_t is the mean absolute deviation of the gradients from μ_t, and s_t^{eff} measures how many scalar gradient values substantially carry the mean-centered gradient magnitude. For a nonzero $\mathbf{h}_t$, $s_t^{\text{eff}} \in [2, d]$, where the lower bound follows from the zero-sum property of the mean-centered gradients. The effective support is large when many gradients have comparable magnitudes and decreases when the gradient magnitudes are concentrated in only a few scalar values in the gradient vectors. Equation 3 shows that this concentration is governed by the ratio σ_t/a_t, rather than by σ_t alone. We therefore consider truncation candidates for which a_t remains comparable. Hence, the observed increase in gradient dispersion indicates that a small subset of extremely large gradients increasingly determines the network parameter update.

It leads to an imbalanced temporal credit (temporal contribution of gradients) to the parameter update when the parameter indices associated with large temporal credit vary across time steps, and when the signs of these gradients are inconsistent across time (Bellec et al., 2020). For a parameter index i, define its mean-centered temporal credit, absolute credit, and temporal concentration as

$$\begin{aligned} c_{t,i} &= w_t h_{t,i}, \qquad \pi_{t,i} = \frac{|c_{t,i}|}{\sum_{j\in\mathcal{T}}|c_{j,i}|}, \qquad \Gamma_i = \sum_{t\in\mathcal{T}} \pi_{t,i}^2, \\ |c_{t^\star,i}| \geq r_i \sum_{\substack{s\in\mathcal{T} \\ j\neq t^\star}} |c_{j,i}| \quad &\Longrightarrow \quad \Gamma_i \geq \left(\frac{r_i}{1+r_i}\right)^2. \end{aligned} \tag{4}$$

Here, $\pi_{t,i}$ is the fraction of the total absolute contribution of a single gradient to network parameter i supplied by timestep t, and $\Gamma_i \in [1/|\mathcal{T}|, 1]$ measures how strongly that contribution is concentrated in particular time steps. These quantities are defined for parameter indices satisfying $\sum_{j\in\mathcal{T}} |c_{j,i}| > 0$. If one timestep contributes at least r_i times the combined contribution of all remaining time steps, Γ_i increases as r_i grows. In this case, the update applied to parameter i is determined primarily by a single temporal path rather than by a balanced aggregation of multiple paths.

The resulting update becomes sensitive to the overlap and sign agreement among the gradient entries with large magnitudes. Let $\bar{\mu} = \sum_{t\in\mathcal{T}} w_t \mu_t$ denote the weighted mean of the gradient entries and let $\rho_{ts} = \langle \mathbf{h}_t, \mathbf{h}_s \rangle / (d\sigma_t\sigma_s)$ denote the normalized alignment between two mean-centered gradients at a time step. The squared norm of the aggregated update is exactly decomposed as

$$\left\|\mathbf{G}\right\|_2^2 = d\bar{\mu}^2 + d\sum_{t\in\mathcal{T}} w_t^2\sigma_t^2 + 2d\sum_{\substack{t,s\in\mathcal{T} \\ t<s}} w_t w_s \sigma_t \sigma_s \rho_{ts}, \tag{5}$$

where $\bar{\mu} = \sum_{t \in \mathcal{T}} w_t \mu_t$ denotes the weighted mean of the gradients, and $\rho_{ts} = \langle \mathbf{h}_t, \mathbf{h}_s \rangle / (d\sigma_t \sigma_s)$ denotes the normalized alignment between the mean-centered gradients at time steps t and s. Here, $\langle \cdot, \cdot \rangle$ denotes the inner product, and $t < s$ ensures that each pair of time steps is counted only once. For nonzero σ_t and σ_s, $\rho_{ts} \in [-1, 1]$, where positive and negative values indicate aligned and opposing mean-centered gradient directions, respectively. In Equation 5, the first term represents the contribution of the gradient, the second term captures the gradient dispersions at each time step, and the third term represents pairwise interactions between gradients at different time steps.

The sensitivity of the aggregated gradient magnitude to the temporal alignment is given by

$$\left| \frac{\partial \|\mathbf{G}\|_2^2}{\partial \rho_{ts}} \right| = 2d w_t w_s \sigma_t \sigma_s. \tag{6}$$

Here, $\partial \|\mathbf{G}\|_2^2 / \partial \rho_{ts}$ measures how sensitively the squared norm of the aggregated gradient changes with respect to the alignment between the gradients at time steps t and s, while the absolute value represents the magnitude of this sensitivity. Equation 6 shows that this sensitivity increases with $\sigma_t \sigma_s$. Accordingly, when the gradients exhibit large dispersion, the same variation in their temporal alignment produces a larger change in the magnitude of the aggregated gradient. Positive alignment therefore induces stronger amplification, whereas negative alignment induces stronger cancellation.

Finally, for an L-smooth training objective F, an update $-\eta \mathbf{G}$ satisfies

$$F(\boldsymbol{\theta} - \eta \mathbf{G}) \leq F(\boldsymbol{\theta}) - \eta \langle \nabla F(\boldsymbol{\theta}), \mathbf{G} \rangle + \frac{L\eta^2}{2} \|\mathbf{G}\|_2^2. \tag{7}$$

Here, $F(\theta)$ denotes the objective function evaluated at the model parameters θ, and $\nabla F(\theta)$ is its gradient with respect to θ. The scalar $\eta > 0$ denotes the learning rate, while $\mathbf{G}$ represents the gradient-based update direction used to update the model parameters. The constant $L > 0$ is the smoothness constant of F, which bounds how rapidly its gradient can change. The notation $\langle \nabla F(\theta) and \mathbf{G} \rangle$ denotes the inner product between the true gradient and the update direction. Accordingly, $F(\theta - \eta \mathbf{G})$ represents the objective value after applying one parameter update with step size η along the direction $-\mathbf{G}$. The term $\frac{L\eta^2}{2} \|\mathbf{G}\|_2^2$ is the penalty associated with the update magnitude.

5 Proposed Mechanism

5.1 Overview

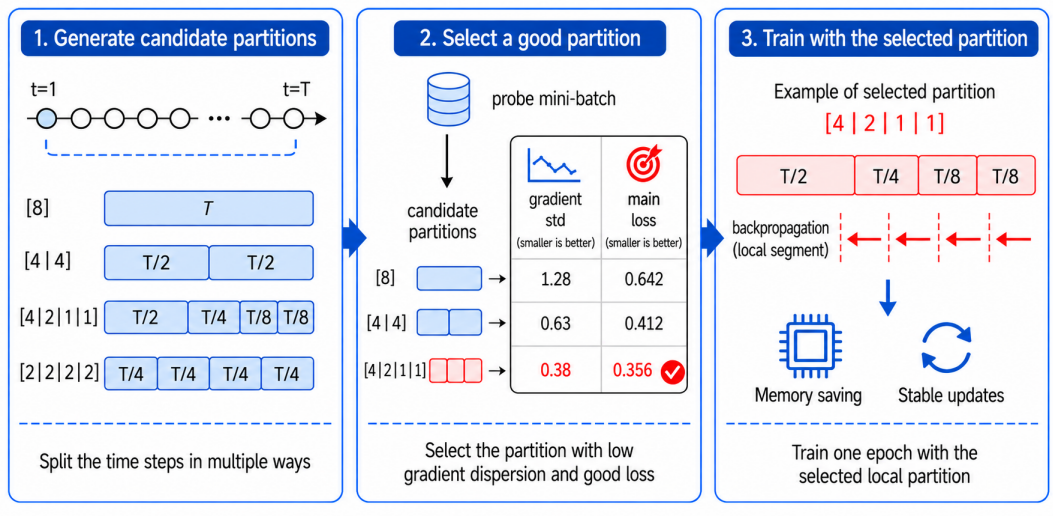

Figure 2: Overall process of LT-BPTT.

As illustrated in Figure 2, LT-BPTT is an adaptive training method that divides the parameter update time into multiple local segments and adjusts this temporal partition during training to reduce gradient dispersion while maintaining task performance. First, LT-BPTT starts with a single partition, as

in normal BPTT, and generates candidate partitions by modifying the partition used in the previous epoch. Second, it evaluates the candidate partitions on the same probe mini-batches using their gradient standard deviation and main task loss, and selects the partition that yields a sufficiently reduced score. Third, it trains the SNN for one epoch using the selected partition, preserving neuronal states across segment boundaries and truncating only the backward gradient paths in time.

5.2 Candidate Partition Generation

LT-BPTT starts from normal BPTT and generates nearby candidate partitions by retaining the current partition, splitting one segment, or merging two adjacent segments. Then, LT-BPTT explores candidate configurations in the neighborhood of the current partitioning configuration and selects the one with the lowest aggregated loss, which is a summation of the standard deviation of gradients and the main task loss. By iteratively repeating this process, LT-BPTT gradually refines the partitioning scheme toward an optimal configuration. While updating the configurations, LT-BPTT removes duplicated and invalid partitioning candidates. Furthermore, we impose a limit on the number of candidates to prevent LT-BPTT's computational complexity from increasing significantly.

5.3 Candidate Partition Evaluation and Selection

LT-BPTT evaluates all candidate partitions using the same probe mini-batches without updating the network parameters. For each candidate, LT-BPTT calculates the gradient standard deviation and the main task loss, where the former favors partitions that produce less dispersed gradients and the latter prevents the search from sacrificing task learning merely to reduce gradient dispersion. Equations 8, 9, and 10 show the losses to optimize the number and lengths of partitions.

$$\mathcal{L}_{\mathrm{std}}(\mathcal{P}) = \frac{1}{QT}\sum_{q=1}^{Q}\sum_{t=1}^{T}\sqrt{\frac{1}{N_g}\sum_{i=1}^{N_g}\left(g_{q,t,i}^{(\mathcal{P})} - \mu_{q,t}^{(\mathcal{P})}\right)^2}. \tag{8}$$

$$\mathcal{L}_{\mathrm{main}}(\mathcal{P}) = \frac{1}{Q}\sum_{q=1}^{Q}\sum_{m=1}^{M}\frac{k_m}{T}\ell_{\mathrm{cls}}\left(\frac{1}{k_m}\sum_{t\in\mathcal{I}_m(\mathcal{P})}\mathbf{z}_{q,t}^{(\mathcal{P})}, \mathbf{y}_q\right). \tag{9}$$

$$\mathcal{J}(\mathcal{P}) = \mathcal{L}_{\mathrm{std}}(\mathcal{P}) + \lambda_{\mathrm{main}}\mathcal{L}_{\mathrm{main}}(\mathcal{P}). \tag{10}$$

Here, $\mathcal{L}_{\mathrm{std}}(\mathcal{P})$[1] is the standard deviation of the gradient, $\mathcal{L}_{\mathrm{main}}(\mathcal{P})$[2] is the main classification loss, and $\mathcal{J}(\mathcal{P})$ is the aggregated loss. We obtain these losses using a candidate partition and a probe mini-batch. $\mathcal{P}$ denotes a candidate partition, while Q and T represent the number of probe mini-batches and the total number of time steps, respectively. $g_{q,t,i}^{(\mathcal{P})}$ denotes the i-th monitored gradient element obtained from the q-th probe mini-batch at time step t, and $\mu_{q,t}^{(\mathcal{P})}$ and N_g denote the mean and the total number of monitored gradient elements, respectively. Moreover, M, k_m, and $\mathcal{I}_m(\mathcal{P})$ represent the number of segments, the length of the m-th segment, and the set of time-step indices included in that segment, respectively. $\mathbf{z}_{q,t}^{(\mathcal{P})}$, $\mathbf{y}_q$, and ℓ_{cls} denote the model logits, ground-truth labels, and classification loss, respectively. Finally, λ_{main} is a scaling factor that controls the contribution of the main classification loss to the partition selection objective (standard deviation). We set λ_{main} to 0.001 for MLP and VGG and 0.01 for ResNet and transformer. The lowest-loss candidate replaces the current one only if its improvement exceeds a threshold.

5.4 Training with the Selected Partition

Once a partition is selected, LT-BPTT uses the same partition for all mini-batches in the subsequent training epoch. Within each mini-batch, the model processes the temporal segments sequentially and

[1] Instead of the main classification loss, we use gradient standard deviation to optimize the numbers and lengths of partitions since it directly reflects how each truncation candidate affects the backward learning process while the main loss only measures current prediction quality of SNNs.

[2] We use Cross-Entropy as the main classification loss.

carries the neuronal states from one segment to the next, while detaching the computational gradient trace at the segment boundary. The losses calculated for the individual segments are weighted by their lengths, and their gradients are accumulated before a single optimizer update. Consequently, different partitions use the same number of updates per mini-batch, while the temporal activation memory is mainly determined by the longest segment rather than the entire simulation length.

6 EXPERIMENTAL RESULTS

We conduct experiments to compare LT-BPTT with the baseline (normal BPTT without mechanisms to improve learning performance) and similar benchmarks to improve the learning performance of SNNs. OTTT enables forward-in-time online learning with constant memory usage, independent of the number of time steps (Xiao et al., 2022a). Temporal Efficient Training (TET) optimizes the pre-synaptic inputs at every time step to help SNNs escape sharp local minima and converge to flatter regions (Deng et al., 2022a). SLTT omits less important temporal backpropagation paths in the computational graph to achieve constant memory usage and faster training (Meng et al., 2023b). A-TBPTT is an adaptive training scheme for RNNs that adjusts the truncation length to control gradient bias based on gradient decay (Aicher et al., 2020). Temporal Model Calibration (TMC) calibrates SNNs and improves performance by dynamically rescaling temporal logit gradients to prevent overconfidence (Yan et al., 2025). Rate-based BP simplifies BPTT into single spatial backpropagation using firing-rate approximations, reducing SNN memory and training time (Yu et al., 2024). To ensure the reliability of the experimental results, we conduct each experiment over 10 independent runs and report the mean performance in the tables.

6.1 CLASSIFICATION ACCURACY COMPARISON

We compare the classification accuracy of SNN models with the benchmarks and the proposed mechanism to evaluate the models' learning performance. In most cases, the models using our mechanism achieve the highest classification accuracy with both 4 and 8 time steps[3].

Table 2: Classification accuracy (%) with 8 time steps across various SNN models integrated with the baseline, benchmarks, and proposed mechanism using diverse datasets.

Dataset	Baseline	TET	OTTT	SLTT	A-TBPTT	TMC	Rate	Proposed
MLP								
MNIST	97.93	**98.25**	**98.25**	98.02	<u>98.14</u>	97.43	97.96	**98.25**
F-MNIST	87.8	<u>89.53</u>	89.26	88.57	89.02	88.92	88.65	**89.77**
N-MNIST	98.03	<u>98.11</u>	98.1	98.02	97.39	97.89	97.38	**98.32**
VGG-11								
CIFAR-10	81.19	83.46	84.27	<u>86.79</u>	84.46	83.58	84.27	**87.98**
DVS-CIFAR-10	63.3	66	63.4	<u>70.1</u>	67.5	68.5	69.2	**71.1**
DVS-Gesture-128	78.12	84.65	80.91	<u>85.76</u>	78.04	81.54	82.83	**85.92**
ResNet-18								
CIFAR-100	70.75	70.93	71.04	68.69	70.14	70.28	<u>71.23</u>	**72.16**
DVS-CIFAR-100	42.67	43.19	43.11	<u>44.35</u>	40.32	41.95	42.74	**44.39**
Caltech-101	67.85	67.84	<u>69.73</u>	65.09	68.48	67.29	68.26	**72.56**
N-Caltech-101	68.65	69.17	70.2	<u>72.44</u>	69.23	68.89	69.35	**73.64**
Transformer with 16 layers								
Tiny-ImageNet	46.53	<u>47.71</u>	45.26	44.85	46.92	46.44	46.98	**49.04**
N-Mini-ImageNet	28.35	<u>29.42</u>	27.08	26.99	29.21	28.52	29.2	**31.37**

Note. Best and second-best available results in each row are **bold** and <u>underlined</u>, respectively; ties are treated equally. Blank entries denote results not yet available.

Table 2 presents the classification accuracy of diverse SNN models with the benchmarks and the proposed mechanism. Among SNN models, models with LT-BPTT exhibit the best learning perfor-

[3]We show the experimental results with 8 time steps in this section. Please refer to the Appendix for the results with 4 time steps

mance and highest classification accuracy. This is because LT-BPTT adaptively restructures temporal credit propagation with the gradient statistics observed during training, unlike the benchmarks. This adaptive partitioning mitigates destructive interactions among heterogeneous temporal credits while retaining temporal information that remains beneficial for gradient updates. We observe that LT-BPTT yields larger accuracy gains when we use temporal datasets. This observation suggests that LT-BPTT is particularly effective when temporal credit is highly heterogeneous. Different from static inputs, temporal sequences contain time-varying information, leading to conflicting per-timestep gradients, for which adaptive temporal partitioning is more beneficial. We also check that among the benchmarks, SLTT achieves the largest improvement in accuracy. This is because it removes inaccurate gradient calculations while retaining the calculation of dominant gradients.

6.2 Memory occupation comparison

We compare the memory occupation of the benchmarks and our mechanism. LT-BPTT significantly reduces the memory occupation of gradient updates compared to the baseline and benchmarks.

Table 3: Memory comparison of SNN training methods with a basic MLP SNN model. "Total" denotes the peak incremental tensor memory estimated over forward, backward, and one optimizer update, while "Back" denotes the peak unique tensor storage for backward computation. Memory values are in MiB and lower is better. Each reduction is computed against full BPTT at the same T using the unrounded measurements.

	$T = 4$				$T = 8$			
Method	Total	Reduction (%)	Back	Reduction (%)	Total	Reduction (%)	Back	Reduction (%)
Baseline	6.1	0	4	0	10.14	0	8	0
TET	6.1	0	4.06	-1.34	10.14	0	8.12	-1.4
OTTT-A	3.57	41.47	1	74.91	3.57	64.79	1	87.45
SLTT	3.41	44.11	1	74.91	3.41	66.37	1	87.45
A-TBPTT	6.1	0	4.02	-0.37	10.14	0	8.04	-0.43
TMC	6.1	0	4.04	-0.92	10.14	0	8.08	-0.95
Rate	3.02	50.45	1.39	65.35	3.02	70.19	1.39	82.66
LT-BPTT, $k_{\max} = 2$	5.31	12.89	2	49.94	5.32	47.54	2	74.95
LT-BPTT, $k_{\max} = T/2$	5.31	12.89	2	49.94	7.63	24.71	4	49.97

Table 3 presents the memory footprint of the MLP SNN model with the baseline, benchmarks, and LT-BPTT. For LT-BPTT, $k_{\max} = 2$ means that the maximum length of each temporal segment is restricted to two time steps, resulting in relatively short truncation intervals. In contrast, $k_{\max} = T/2$ sets the maximum temporal segment length to half of the total simulation length T, allowing longer temporal dependencies to be preserved within each segment. For example, when $T = 8$, $k_{\max} = 2$ corresponds to a partition such as $(2, 2, 2, 2)$, whereas $k_{\max} = T/2 = 4$ corresponds to $(4, 4)$. LT-BPTT substantially reduces the memory required for gradient updates compared with BPTT, achieving reductions of 47.54% in peak tensor memory and 74.95% in backward storage at $T = 8$ with $k_{\max} = 2$. In contrast, TET and TMC retain memory requirements comparable to BPTT, while A-TBPTT provides no reduction in peak memory. Although OTTT-A, SLTT, and Rate achieve smaller memory footprints through temporal decoupling or gradient compression, LT-BPTT preserves temporal gradient updates within local segments while substantially reducing its storage cost. These fixed-partition results demonstrate that LT-BPTT offers a configurable trade-off between the length of retained temporal credits and memory occupation.

7 Conclusion

In this paper, we investigated how temporal truncation affects gradient behavior during SNN training. Based on our findings, we proposed LT-BPTT, an adaptive training framework that jointly determines the number and lengths of temporal segments. LT-BPTT generates candidate partitions, evaluates them using gradient dispersion, and progressively selects effective temporal partitions during training. In this way, LT-BPTT produces more stable and effective gradient updates, while reducing memory occupation through shorter backpropagation paths. Extensive experiments across diverse SNN architectures and datasets demonstrated that LT-BPTT achieves the highest classification accuracy in most evaluated settings while reducing memory consumption compared with conventional BPTT. Overall, our study shows that adaptively controlling temporal truncation in gradient updates can simultaneously improve SNNs' learning performance and training efficiency.

AI USAGE STATEMENT

We used AI agents as writing, experimental, and figure assistants to improve the clarity of writing/editing mathematical equations (fixing typos/suggesting algebraic simplifications) and assist us in conducting experiments with the benchmarks. We did not use LLMs for idea generation, methodological design, and analysis of experimental results. We derived, verified, and finalized all derivations, results, and claims.

REPRODUCIBILITY STATEMENT

We conducted all experiments on publicly available datasets with standard train/validation/test splits. To facilitate replication, we provide our full implementation of the proposed mechanisms in the anonymous supplementary material. We also present the hyperparameter settings, model configurations, and hardware specifications to support reproducibility. We will publicly release the code and scripts on GitHub if our paper is accepted for the conference.

REFERENCES

Christopher Aicher, Nicholas J. Foti, and Emily B. Fox. Adaptively truncating backpropagation through time to control gradient bias. In *Proceedings of the 35th Conference on Uncertainty in Artificial Intelligence*, volume 115 of *Proceedings of Machine Learning Research*, pp. 799–808. PMLR, 2020. URL https://proceedings.mlr.press/v115/aicher20a.html.

Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo Di Nolfo, Tapan K. 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. doi: 10.1109/CVPR.2017.781.

Guillaume Bellec, Franz Scherr, Anand Subramoney, Elias Hajek, Darjan Salaj, Robert Legenstein, and Wolfgang Maass. A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature Communications*, 11(1):3625, 2020. doi: 10.1038/s41467-020-17236-y.

Tong Bu, Wei Fang, Jianhao Ding, Penglin Dai, Zhaofei Yu, and Tiejun Huang. Optimal ann-snn conversion for high-accuracy and ultra-low-latency spiking neural networks. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=7B3IJMM1k_M.

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*, 2022a.

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*, 2022b. URL https://openreview.net/forum?id=_XNtisL32jv.

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

Jason K. Eshraghian, Max Ward, Emre O. Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D. Lu. Training spiking neural networks using lessons from deep learning. *Proceedings of the IEEE*, 111(9):1016–1054, 2023. doi: 10.1109/JPROC.2023.3308088.

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

Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 106(1):59–70, 2007. doi: 10.1016/j.cviu.2005.09.012.

Wulfram Gerstner and Werner M. Kistler. *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press, Cambridge, UK, 2002. doi: 10.1017/CBO9780511815706.

Wenzhe Guo, Mohammed E. Fouda, Ahmed M. Eltawil, and Khaled N. Salama. Efficient training of spiking neural networks with temporally-truncated local backpropagation through time. *Frontiers in Neuroscience*, 17:1047008, 2023. doi: 10.3389/fnins.2023.1047008.

Julia Gygax and Friedemann Zenke. Elucidating the theoretical underpinnings of surrogate gradient learning in spiking neural networks. *Neural Computation*, 37(5):886–925, 2025. doi: 10.1162/neco_a_01752.

Zecheng Hao, Tong Bu, Jianhao Ding, Tiejun Huang, and Zhaofei Yu. Reducing ann-snn conversion error through residual membrane potential. In *Proceedings of the AAAI Conference on Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press, 2023. ISBN 978-1-57735-880-0. doi: 10.1609/aaai.v37i1.25071. URL https://doi.org/10.1609/aaai.v37i1.25071.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.

Yechan Kang, Yongjin Kweon, Mingyeong Seo, Sohee Park, Yeonguk Jeon, Jongkil Park, Hyun Jae Jang, Jaewook Kim, YeonJoo Jeong, Suyoun Lee, and Seongsik Park. A2SG: Adaptive and asymmetric surrogate gradients for training deep spiking neural networks. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, 2026.

Junho Kim, Jaehyeok Bae, Gangin Park, Dongsu Zhang, and Young Min Kim. N-imagenet: Towards robust, fine-grained object recognition with event cameras. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2146–2156, October 2021.

Vijay Anand Korthikanti, Jared Casper, Sangkug Lym, Lawrence McAfee, Michael Andersch, Mohammad Shoeybi, and Bryan Catanzaro. Reducing activation recomputation in large transformer models. In *Proceedings of Machine Learning and Systems*, volume 5, 2023. URL https://proceedings.mlsys.org/paper_files/paper/2023/hash/80083951326cf5b35e5100260d64ed81-Abstract-mlsys2023.html.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. URL https://cave.cs.toronto.edu/kriz/learning-features-2009-TR.pdf.

Ya Le and Xuan Yang. Tiny ImageNet visual recognition challenge. CS231n course project report, Stanford University, 2015. URL https://cs231n.stanford.edu/reports/2015/pdfs/yle_project.pdf.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

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. doi: 10.3389/fnins.2017.00309.

Yuhang Li, Shikuang Deng, Xin Dong, Ruihao Gong, and Shi Gu. A free lunch from ann: Towards efficient, accurate spiking neural networks calibration. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 6316–6325. PMLR, 2021a. URL https://proceedings.mlr.press/v139/li21d.html.

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, 2021b.

Shuang Lian, Jiangrong Shen, Qianhui Liu, Ziming Wang, Rui Yan, and Huajin Tang. Learnable surrogate gradient for direct training spiking neural networks. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pp. 3002–3010, 2023. doi: 10.24963/ijcai.2023/335.

Xu Liu, Na Xia, Jinxing Zhou, Jingyuan Xu, and Dan Guo. A closer look at knowledge distillation in spiking neural network training. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(3):1946–1954, 2026. doi: 10.1609/aaai.v40i3.37175.

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

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, 2023a.

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 (ICCV)*, pp. 6166–6176, 2023b.

Satoshi Moriya, Masaya Ishikawa, Satoshi Ono, Hideaki Yamamoto, Yasushi Yuminaka, Yoshihiko Horio, Jordi Madrenas, and Shigeo Sato. Analog vlsi implementation of subthreshold spiking neural networks and its application to reservoir computing. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 72(10):5571–5582, 2025. doi: 10.1109/TCSI.2025.3550876.

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

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. doi: 10.3389/fnins.2015.00437.

Christian Pehle, Sebastian Billaudelle, Benjamin Cramer, Jakob Kaiser, Korbinian Schreiber, Yannik Stradmann, Johannes Weis, Aron Leibfried, Eric Müller, and Johannes Schemmel. The BrainScaleS-2 accelerated neuromorphic system with hybrid plasticity. *Frontiers in Neuroscience*, 16:795876, 2022. doi: 10.3389/fnins.2022.795876.

Corentin Tallec and Yann Ollivier. Unbiasing truncated backpropagation through time. *arXiv preprint arXiv:1705.08209*, 2017. URL `https://arxiv.org/abs/1705.08209`.

Clarence Tan, Marko Šarlija, and Nikola Kasabov. Spiking neural networks: Background, recent development and the neucube architecture. *Neural Processing Letters*, 52(2):1675–1701, 2020. doi: 10.1007/s11063-020-10322-8.

Ziming Wang, Runhao Jiang, Shuang Lian, Rui Yan, and Huajin Tang. Adaptive smoothing gradient learning for spiking neural networks. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 35798–35816. PMLR, 2023. URL `https://proceedings.mlr.press/v202/wang23j.html`.

Paul J. Werbos. Backpropagation through time: What it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990. doi: 10.1109/5.58337.

Ronald J. Williams and Jing Peng. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Computation*, 2(4):490–501, 1990. doi: 10.1162/neco.1990.2.4.490.

Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in Neuroscience*, 12:331, 2018. doi: 10.3389/fnins.2018.00331.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms, 2017. URL https://arxiv.org/abs/1708.07747.

Mingqing Xiao, Qingyan Meng, Zongpeng Zhang, Di He, and Zhouchen Lin. Online training through time for spiking neural networks. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 20717–20730. Curran Associates, Inc., 2022a.

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, 2022b.

Jiaqi Yan, Changping Wang, De Ma, Huajin Tang, Qian Zheng, and Gang Pan. Training high performance spiking neural network by temporal model calibration. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=l7ZmdeFyM1.

Man Yao, Xuerui Qiu, Tianxiang Hu, Jiakui Hu, Yuhong Chou, Keyu Tian, Jianxing Liao, Luziwei Leng, Bo Xu, and Guoqi Li. Scaling spike-driven transformer with efficient spike firing approximation training. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(4): 2973–2990, 2025. doi: 10.1109/TPAMI.2025.3530246.

Kangning Yin, Zhen Ding, Rui Zhu, Shaoqi Hou, Ye Li, Yan Wang, and Yujian Du. Fedspike: A personalized federated learning method for energy-efficient spiking neural networks in edge intelligence. *Information Sciences*, 755:123785, 2026. doi: 10.1016/j.ins.2026.123785.

Chengting Yu, Lei Liu, Gaoang Wang, Erping Li, and Aili Wang. Advancing training efficiency of deep spiking neural networks through rate-based backpropagation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=wlcm21C4nk.

Friedemann Zenke and Surya Ganguli. Superspike: Supervised learning in multilayer spiking neural networks. *Neural Computation*, 30(6):1514–1541, 2018. doi: 10.1162/neco_a_01086.

Tianqing Zhang, Kairong Yu, Xian Zhong, Hongwei Wang, Qi Xu, and Qiang Zhang. Staasnn: Spatial-temporal attention aggregator for spiking neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13959–13969, 2025. doi: 10.1109/CVPR52734.2025.01303.

Hanle Zheng, Yujie Wu, Lei Deng, Yifan Hu, and Guoqi Li. Going deeper with directly-trained larger spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11062–11070, 2021. doi: 10.1609/aaai.v35i12.17320.

Chenlin Zhou, Han Zhang, Liutao Yu, Yumin Ye, Zhaokun Zhou, Liwei Huang, Zhengyu Ma, Xiaopeng Fan, Huihui Zhou, and Yonghong Tian. Direct training high-performance deep spiking neural networks: A review of theories and methods. *Frontiers in Neuroscience*, 18:1383844, 2024. doi: 10.3389/fnins.2024.1383844.

APPENDIX INDEX

Appendix A Experimental settings

Appendix B Experimental results with additional configurations

- **B.1** Fewer time steps
- **B.2** Deeper SNN models
- **B.3** In-depth study
  - **B.1.1** Relation between gradient dispersion and loss curves
  - **B.1.2** Ablation on the composition of LT-BPTT

Appendix C Efficiency comparison

- **C.1** Time and space complexity
- **C.2** Energy consumption on GPU
- **C.3** Processing time on GPU

Appendix D Implementation with a neuromorphic device

- **D.1** Performance evaluation
- **D.2** Energy consumption on the neuromorphic device
- **D.3** Processing time on the neuromorphic device

Appendix E Theoretical analysis of the partition selection objective

- **E.1** Analysis setting and gradient notation
  - **E.1.1** A common forward computation
  - **E.1.2** Monitored gradients and the training update
  - **E.1.3** Averages over time steps
- **E.2** How gradient dispersion bounds temporal credit aggregation
  - **E.2.1** A direct bound using the implemented loss
  - **E.2.2** What these bounds do not imply
- **E.3** How the main loss measures prediction disagreement
  - **E.3.1** Predictions for each example
  - **E.3.2** Full-sequence loss and disagreement
- **E.4** What an accepted partition guarantees
  - **E.4.1** The objective used for candidate comparison
  - **E.4.2** Splitting and merging have different effects
- **E.5** How the objective bounds the one-step training loss
  - **E.5.1** Separating segment supervision from temporal truncation
  - **E.5.2** The additional discrepancy caused by truncation
  - **E.5.3** When the reference loss is guaranteed to decrease
- **E.6** Scope for SNN training and implementation
  - **E.6.1** A gradient decomposition consistent with the segment loss
  - **E.6.2** Why truncation discrepancy remains a separate condition
  - **E.6.3** Hard spikes and a local differentiable reference
  - **E.6.4** Monitored coordinates and optimizer transformations
  - **E.6.5** What the analysis establishes

A EXPERIMENTAL SETTINGS

Table 4: Configurations of each SNN model in our experiments.

	MLP	VGG	RESNET	TRANSFORMER
DATASETS	MNIST/F-/N-	CIFAR-10/DVS-DVS-GESTURE-128	CIFAR-100/DVS-CALTECH-101/N-	TINY-IMAGENET/MINI-N-
NEURON MODELS	LIF	LIF	IF	IF
MAIN LOSS FUNCTION	CE	CE	CE	CE
BATCH SIZE	128	128/32	128/32	256/128
OPTIMIZER	ADAM	ADAM	SGD	ADAM
LEARNING RATE	0.001	0.001	0.1	0.01
DEPTHS	5	11/15	18/34	16
NUMBER OF TIME STEPS	4/8	4/8	4/8	4/8
NUMBER OF TRAINING EPOCHS	50	1.00	300	300

Table 4 shows the configuration settings for our experimental results. We use MNIST, F-MNIST, and N-MNIST for the MLP model. We use CIFAR-10/100, Caltech-101, and Tiny-ImageNet for VGG/ResNet and the spiking transformer (LeCun et al., 1998; Xiao et al., 2017; Orchard et al., 2015; Krizhevsky, 2009; Fei-Fei et al., 2007; Le & Yang, 2015). We also use DVS-Gesture-128 and apply the DVS or neuromorphic versions of CIFAR-10/100, Caltech-101, and Tiny-ImageNet to VGG/ResNet and the spiking transformer (Amir et al., 2017; Li et al., 2017; Orchard et al., 2015; Liu et al., 2026; Kim et al., 2021). We adopt Cross-Entropy (CE) as a loss function since we use classification datasets. We apply early stopping with a patience of 10 to SNN models during training. LIF neurons for MLP/VGG and IF neurons for ResNet/transformer. This is because leakage of LIF neurons attenuates the propagation of information and gradients along the temporal dimension. In addition, we set configurations for our mechanism through repeated experiments. To estimate the training memory occupation, we use a controlled FP32 LIF-MLP, a batch size of 128, and simulation lengths of $T \in \{4, 8\}$. We measure the peak tensor memory allocated during forward propagation, backward propagation, and one optimizer update (Korthikanti et al., 2023). In addition, we separately measure the peak tensor storage for backward computation to isolate the memory cost associated with temporal gradient propagation. We use Ubuntu 24.04 servers with RTX 4080, RTX 5080, and RTX 5070 Ti for our experiments. Furthermore, we implement SNN models integrated with the benchmarks and the proposed mechanism on BrainScaleS-2, a widely used neuromorphic device (Pehle et al., 2022).

B EXPERIMENTAL RESULTS WITH ADDITIONAL CONFIGURATIONS

B.1 FEWER TIME STEPS

Table 5: Classification accuracy (%) with 4 time steps across various SNN models integrated with the baseline, benchmarks, and proposed mechanism using diverse datasets.

Dataset	Baseline	TET	OTTT	SLTT	A-TBPTT	TMC	Rate	Proposed
MLP								
MNIST	98.05	97.99	98.08	98.05	98.12	97.31	**98.26**	98.2
F-MNIST	89.46	89.81	89.73	88.39	89.02	89.14	89.14	**89.94**
N-MNIST	97.19	97.34	97.27	97.56	97.39	97.38	97.26	**97.79**
VGG-11								
CIFAR-10	80.25	83.3	83.86	86.36	83.89	82.35	81.95	**86.51**
DVS-CIFAR-10	58.4	65.1	61.3	66.5	59.63	62.8	61.49	**68.6**
DVS-Gesture-128	73.96	74.65	74.67	78.13	76.74	75.54	74.97	**83.33**
ResNet-18								
CIFAR-100	69.89	70.32	70.68	68.69	64.13	69.53	70.64	**71.05**
DVS-CIFAR-100	40.48	40.92	40.56	40.78	36.2	39.3	39.6	**41.19**
Caltech-101	64.87	65.73	65.31	68.1	67.45	65.94	64.65	**68.83**
N-Caltech-101	60.62	62.57	61.68	66.44	65.56	66.1	65.97	**67.29**
Transformer with 16 layers								
Tiny-ImageNet	42.68	44.83	42.77	41.09	43.04	42.92	43.15	**45.56**
N-Mini-ImageNet	25.58	26.04	25.93	23.91	26.12	25.74	26.27	**28.12**

Note. Best and second-best available results in each row are **bold** and underlined, respectively; ties are treated equally. Blank entries denote results not yet available.

Table 5 presents the classification accuracy of SNN models using the baseline, benchmarks, and LT-BPTT with 4 time steps. LT-BPTT achieves the highest classification accuracy with 11 evaluated datasets and obtains the second-best result with MNIST. The improvement is particularly pronounced on neuromorphic datasets, reaching 83.33% on DVS-Gesture-128 and 68.60% on DVS-CIFAR-10, outperforming the best competing methods by 5.2% and 2.1%, respectively. Overall, LT-BPTT consistently improves the learning performance across MLP, VGG-11, ResNet-18, and Transformer architectures, indicating that its adaptive temporal truncation remains effective across diverse network structures and data modalities.

B.2 DEEPER SNN MODELS

Table 6: Classification accuracy (%) with 8 time steps across deeper SNN models integrated with the baseline, benchmarks, and proposed mechanism using diverse datasets.

Dataset	Baseline	TET	OTTT	SLTT	A-TBPTT	TMC	Rate	Proposed
VGG-15								
CIFAR-10	79.72	82.43	84.05	**85.66**	82.94	82.61	83.73	85.58
DVS-CIFAR-10	57.3	54	57.2	65.3	63.5	60.7	64.6	**66.3**
DVS-Gesture-128	77.08	72.91	77.63	83.17	76.89	80.29	80.78	**84.69**
ResNet-34								
CIFAR-100	69.47	71.69	70.62	69.71	69.19	70.59	71.13	**72.08**
DVS-CIFAR-100	41.92	31.36	38.83	38.12	41.34	41.73	42.37	**42.45**
Caltech-101	67.63	57.83	68.49	71.49	68.17	67.54	67.98	**72.31**
N-Caltech-101	69.12	66.59	70.07	73.82	69.61	69.29	69.52	**74.77**

Note. Best and second-best available results in each row are **bold** and underlined, respectively; ties are treated equally. Blank entries denote results not yet available.

Table 6 shows the classification accuracy of SNN models integrated with the baseline, benchmarks, and proposed mechanism. We observe that the overall classification accuracy of the models decreases. This is because sparse binary activations by spiking neurons and repeated surrogate-gradient

operations attenuate effective gradient updates across layers. Despite this degradation, LT-BPTT achieves the greatest improvement in the models' learning performance.

B.3 In-depth study

B.3.1 Relation between gradient dispersion and learning performance

We conduct experiments to investigate the relation between gradient dispersion and learning performance. We observe that as the gradient dispersion decreases by LT-BPTT, the classification accuracy of SNN models improves.

Table 7: Evolution of the gradient dispersion and training accuracy during LT-BPTT training. The gradient dispersion consistently decreases as training accuracy improves across different network architectures.

Model	**Metric**	**Ep. 10**	**Ep. 20**	**Ep. 30**	**Ep. 40**	**Ep. 50**
MLP	Gradient dispersion ↓	5.68×10^{-5}	3.83×10^{-5}	2.36×10^{-5}	8.57×10^{-6}	4.11×10^{-6}
	Train Acc. (%) ↑	99.72	99.94	99.94	100.00	100.00
VGG-11	Gradient dispersion ↓	1.27×10^{-4}	9.44×10^{-5}	7.60×10^{-5}	6.80×10^{-5}	6.37×10^{-5}
	Train Acc. (%) ↑	68.24	77.87	88.53	89.48	94.54
ResNet-18	Gradient dispersion ↓	2.98×10^{-4}	2.62×10^{-4}	2.58×10^{-4}	2.55×10^{-4}	2.44×10^{-4}
	Train Acc. (%) ↑	50.14	74.45	81.91	93.49	99.29
Transformer	Gradient dispersion ↓	8.14×10^{-4}	6.93×10^{-4}	6.65×10^{-4}	6.17×10^{-4}	5.81×10^{-4}
	Train Acc. (%) ↑	18.68	26.95	33.83	39.74	45.12

Table 7 presents the evolution of the gradient dispersion and training accuracy of LT-BPTT training. Across all SNN models, the gradient dispersion consistently decreases as training progresses. At the same time, the training accuracy increases for MLP, VGG, ResNet, and Transformer SNN models. These results show a consistent association between reduced gradient dispersion and improved learning performance caused by LT-BPTT training.

B.3.2 Ablation on the composition of LT-BPTT

To analyze the increase in learning performance attributable to LT-BPTT, we conduct an ablation study by adopting the standard deviation and main loss, respectively.

Table 8: Classification accuracy (%) of various SNN models comparing the individual and combined objective components across different architectures and datasets.

Architecture (Dataset)	STD only	Main only	**Proposed (STD + main)**
MLP (MNIST)	98.19	98.10	**98.25**
VGG-11 (CIFAR-10)	83.08	82.27	**87.98**
ResNet-18 (CIFAR-100)	71.93	71.54	**72.16**
Transformer (Tiny-ImageNet)	48.65	47.86	**49.04**

Table 8 presents the classification accuracy of various SNN models with LT-BPTT as the composition of the losses changes. The proposed objective, combining the standard deviation of gradients with the main task loss, consistently achieves the highest accuracy across all SNN models and datasets. The improvement is particularly substantial for VGG-11 on CIFAR-10, where the combined objective reaches 87.98%, compared with 83.08% and 82.27% when using each term individually. These results indicate that the two objective components provide complementary benefits and are both important for effective LT-BPTT optimization.

C Efficiency comparison

We theoretically calculate the time and space complexity of the baseline, benchmarks, and proposed mechanism. Furthermore, we measure the energy consumption and processing time of SNN models on an RTX 5080 GPU. The SNN models with LT-BPTT exhibit smaller energy consumption and shorter processing time than the models with the baseline and benchmarks.

C.1 TIME AND SPACE COMPLEXITY

To compare the overhead of the baseline, benchmarks, and LT-BPTT, we calculate their time and space complexities (Werbos, 1990).

Table 9: Notation used in the complexity analysis.

Symbol	Definition
T	Number of simulation time steps
P	Number of trainable parameters
A	Activation/state storage for one time step
A_f	Working-tensor storage for one no-gradient forward time step
$\mathcal{F}$	Forward computation cost for one time step
$\mathcal{B}$	Spatial backward computation cost for one time step
H	Number of output elements per time step ($H = bC$)
K	Maximum temporal partition length in LT-BPTT
K_{A}	Adaptive truncation length in A-TBPTT

Table 9 presents the notations of our complexity equations. T and P denote the numbers of simulation time steps and trainable parameters, respectively, while A, A_f, $\mathcal{F}$, and $\mathcal{B}$ represent the storage and computational costs per time step. H denotes the number of output elements per time step, K is the maximum temporal partition length used in LT-BPTT, and K_{A} is the adaptive truncation length of A-TBPTT.

Table 10: Space and time complexity of the compared SNN training methods.

Method	Time Complexity	Space Complexity
BPTT	$\mathcal{O}(T(\mathcal{F}+\mathcal{B})+P)$	$\mathcal{O}(P+TA)$
TET	$\mathcal{O}(T(\mathcal{F}+\mathcal{B})+TH+P)$	$\mathcal{O}(P+TA)$
OTTT	$\mathcal{O}(T(\mathcal{F}+\mathcal{B}+A)+P)$	$\mathcal{O}(P+A)$
SLTT	$\mathcal{O}(T(\mathcal{F}+\mathcal{B})+P)$	$\mathcal{O}(P+A)$
A-TBPTT	$\mathcal{O}(T(\mathcal{F}+\mathcal{B})+U_{\mathrm{A}}P)$	$\mathcal{O}(P+K_{\mathrm{A}}A)$
TMC	$\mathcal{O}\big(T(\mathcal{F}+\mathcal{B})+T^2H+P\big)$	$\mathcal{O}(P+TA)$
Rate	$\mathcal{O}((T+1)\mathcal{F}+\mathcal{B}+TA+P)$	$\mathcal{O}(P+A+TA_f)$
LT-BPTT	$\mathcal{O}(T(\mathcal{F}+\mathcal{B})+P)$	$\mathcal{O}(P+KA)$

Table 10 shows the time and space complexities of the baseline, benchmark, and proposed mechanism. BPTT, TET, TMC, and Rate require memory that scales with the total simulation length T. In contrast, LT-BPTT reduces the space complexity to $\mathcal{O}(P+KA)$. Its time complexity remains $\mathcal{O}(T(\mathcal{F}+\mathcal{B})+P)$, without the additional overhead terms required by several competing methods. Therefore, LT-BPTT achieves a favorable complexity profile by reducing memory usage while maintaining low computational cost.

C.2 ENERGY CONSUMPTION ON GPU

Table 11: Energy consumption (J) to process a single mini-batch with a size of 32 (DVS-CIFAR-10 and DVS-Gesture-128) and 128 (other datasets) across various SNN models integrated with the baseline, benchmarks, and proposed mechanism using 8 time steps.

Dataset	Baseline	TET	OTTT	SLTT	A-TBPTT	TMC	Rate	Proposed
MLP								
MNIST	0.52	0.54	1.21	0.87	0.85	0.55	0.61	0.69
F-MNIST	0.50	0.53	1.22	0.88	0.87	0.56	0.68	0.71
N-MNIST	1.98	1.99	2.79	2.33	2.32	1.99	2.24	2.16
VGG-11								
CIFAR-10	25.36	25.64	22.13	19.65	25.92	25.68	26.83	21.27
DVS-CIFAR-10	87.91	88.23	71.02	77.45	90.22	90.08	90.97	84.68
DVS-Gesture-128	313.49	316.01	255.48	297.14	321.47	323.82	327.55	302.91
ResNet-18								
CIFAR-100	71.91	71.83	71.50	50.20	72.69	71.89	71.73	55.73
DVS-CIFAR-100	98.97	99.53	84.75	65.40	98.99	99.05	98.99	71.46
Caltech-101	100.66	101.53	93.02	66.87	99.95	100.66	99.47	73.45
N-Caltech-101	81.76	83.20	76.61	62.85	81.57	83.17	82.29	65.10
Transformer with 16 layers								
Tiny-ImageNet	75.81	76.12	65.21	56.83	75.63	75.87	75.79	56.93
N-Mini-ImageNet	70.08	70.32	61.19	47.42	69.98	70.16	70.13	47.28

Table 11 presents the energy consumption of various SNN models with the baseline, benchmarks, and proposed mechanism. LT-BPTT generally achieves the second- or third-lowest energy consumption for the MLP, VGG, ResNet, and Transformer SNN models. In particular, its energy consumption remains close to the most energy-efficient methods while being substantially lower than the baseline in most of these settings. Although LT-BPTT is not the most energy-efficient method for every dataset, it maintains a favorable overall energy profile across diverse network architectures. This efficiency mainly comes from restricting temporal backpropagation to selected temporal partitions, thereby reducing unnecessary gradient propagation and associated computations compared with the baseline and benchmarks. These results indicate that LT-BPTT provides an effective trade-off between training energy efficiency and learning performance.

C.3 PROCESSING TIME ON GPU

Table 12: Processing time (m sec) to process a single mini-batch with a size of 32 (DVS-CIFAR-10 and DVS-Gesture-128) and 128 (other datasets) across various SNN models integrated with the baseline, benchmarks, and proposed mechanism using 8 time steps.

Dataset	Baseline	TET	OTTT	SLTT	A-TBPTT	TMC	Rate	Proposed
MLP								
MNIST	9.28	9.65	19.00	15.39	14.18	9.54	10.67	12.16
F-MNIST	8.67	9.19	19.25	15.32	14.39	9.76	11.02	12.41
N-MNIST	39.09	39.13	48.16	44.60	43.11	39.23	43.18	41.93
VGG-11								
CIFAR-10	115.81	115.91	99.21	85.30	118.41	116.79	120.59	95.64
DVS-CIFAR-10	408.64	412.76	349.38	365.98	419.40	414.43	426.53	393.00
DVS-Gesture-128	1518.16	1532.74	1265.99	1418.06	1589.39	1605.51	1624.30	1479.08
ResNet-18								
CIFAR-100	296.10	296.13	277.68	182.87	300.19	297.53	293.52	214.36
DVS-CIFAR-100	433.63	444.91	401.98	297.27	428.89	434.24	433.61	319.61
Caltech-101	456.82	465.39	432.88	321.12	452.39	453.66	452.58	347.13
N-Caltech-101	650.29	645.97	638.80	550.31	615.43	649.68	650.35	537.18
Transformer with 16 layers								
Tiny-ImageNet	290.28	286.05	266.56	213.88	293.26	291.89	290.56	195.69
N-Mini-ImageNet	251.75	249.36	241.44	180.67	252.72	252.13	251.92	176.81

Table 12 shows the processing time of various SNN models with the baseline, benchmarks, and proposed mechanism. LT-BPTT also exhibits the second- or third-shortest processing time for the MLP, VGG, ResNet, and Transformer SNN models. This is because limiting temporal backpropagation to selected temporal partitions makes the models avoid part of the computational overhead required by BPTT training. LT-BPTT is not always the fastest method because it retains more temporal credit paths than aggressive approximations such as SLTT. Nevertheless, LT-BPTT achieves the strongest overall classification performance in the corresponding accuracy comparison. These results indicate that LT-BPTT provides an effective balance between short training-time processing and improved learning performance.

D IMPLEMENTATION ON A NEUROMORPHIC DEVICE

To demonstrate that LT-BPTT successfully improves the learning performance of real neuromorphic devices, we implement simple MLP SNN models on BrainScaleS-2, a widely used platform for neuromorphic computing (Pehle et al., 2022). We directly train the MLP SNNs on BrainScaleS-2 and report the resulting classification accuracy. Our experimental results show that LT-BPTT enhances the learning performance of the neuromorphic device without significant increases in energy consumption and processing time. We use 32 time steps and mini-batches with a size of 64 for evaluation using the BrainScaleS-2.

D.1 PERFORMANCE EVALUATION

Table 13: Classification accuracy (%) for MLP SNNs with a hidden layer of 64 neurons on BrainScaleS-2.

Dataset	Baseline	TET	OTTT	SLTT	A-TBPTT	TMC	Rate	Proposed
MNIST	95.51	95.78	<u>95.89</u>	95.63	95.72	95.12	95.66	**95.92**
F-MNIST	82.54	<u>84.28</u>	83.94	83.28	83.76	83.68	83.37	**84.53**
N-MNIST	95.67	<u>95.72</u>	95.78	95.69	95.08	95.51	95.02	**95.99**

Note. Best and second-best available results in each row are **bold** and <u>underlined</u>, respectively; ties are treated equally. Blank entries denote results not yet available.

Table 13 presents the classification accuracy of a simple MLP SNN model integrated with the baseline, benchmarks, and proposed mechanism on BrainScaleS-2. LT-BPTT also achieves the best improvement in learning performance of SNNs. This result suggests that our mechanism works well on real neuromorphic devices.

D.2 ENERGY CONSUMPTION ON THE NEUROMORPHIC DEVICE

Table 14: Estimated energy consumption (mJ) using 32 time steps across MLP SNN models and datasets on BrainScaleS-2 to process a single mini-batch with a size of 64.

Dataset	Baseline	TET	OTTT	SLTT	A-TBPTT	TMC	Rate	Proposed
MNIST	90.48	89.86	92.94	92.25	89.81	90.84	89.79	90.51
F-MNIST	90.51	89.89	93.02	92.28	89.87	90.91	89.82	90.56
N-MNIST	90.52	90.29	94.66	92.85	89.89	91.44	90.54	90.88

Table 14 shows the energy consumption of the MLP model using the baseline, benchmarks, and proposed mechanism for backwarding a single mini-batch and updating gradients on BrainScaleS-2. LT-BPTT shows energy consumption comparable to the baseline and other benchmark methods across all datasets. On MNIST and F-MNIST, LT-BPTT consumes $90.53\mathrm{mJ}$, which is nearly identical to the baseline consumption of $90.51\mathrm{mJ}$. On N-MNIST, LT-BPTT requires $90.88\mathrm{mJ}$, remaining within the range observed for the compared methods. These results indicate that LT-BPTT introduces no substantial additional energy overhead on the neuromorphic device.

D.3 PROCESSING TIME ON THE NEUROMORPHIC DEVICE

Table 15: Estimated processing time (ms) using 32 time steps across MLP SNN models and datasets on BrainScaleS-2 to process a single mini-batch with a size of 64 under Scenario A.

Dataset	Baseline	TET	OTTT	SLTT	A-TBPTT	TMC	Rate	Proposed
MNIST	452.49	449.43	465.01	461.35	449.13	454.42	449.06	452.62
F-MNIST	452.57	449.46	465.08	461.37	449.16	454.44	449.1	452.69
N-MNIST	452.58	451.43	473.31	464.27	449.46	457.19	452.71	454.4

Table 15 shows the processing time of MLP SNN integrated with the baseline, benchmarks, and proposed mechanism on BrainScaleS-2 for backwarding a single mini-batch and updating gradients on BrainScaleS-2. The processing time of LT-BPTT is also comparable to those of the baseline and other benchmark methods. On MNIST and F-MNIST, LT-BPTT requires $452.63\mathrm{ms}$, only $0.1\mathrm{ms}$ longer than the baseline. On N-MNIST, LT-BPTT requires $454.4\mathrm{ms}$, which remains within the

range observed for the compared methods. Therefore, LT-BPTT introduces only marginal latency overhead while maintaining comparable efficiency in processing time.

E THEORETICAL ANALYSIS OF THE PARTITION SELECTION OBJECTIVE

We connect the temporal credit analysis in Equations 5 and 7 to the partition selection objective in Equation 10. We establish three main results. First, the gradient standard deviation bounds the magnitude of the mean-centered aggregated gradient and its sensitivity to temporal alignment. Second, the main classification loss equals the full-sequence classification loss plus a penalty for disagreement between segment predictions. Third, under explicit local smoothness and gradient-discrepancy conditions, the combined objective bounds the average one-step training loss. The first two results follow from algebra and do not require a smooth spike function. The analysis uses the losses in Equations 8 and 9 without changing the partition selection rule.

E.1 ANALYSIS SETTING AND GRADIENT NOTATION

We compare candidates at the same network parameters and explicitly connect the monitored gradients to the accumulated training gradient.

E.1.1 A COMMON FORWARD COMPUTATION.

Let $\mathcal{C}$ be the finite set of candidate partitions evaluated at the current network parameters $\boldsymbol{\theta}$. The indices q, t, and m, the segment length k_m, and the segment index set $\mathcal{I}_m(\mathcal{P})$ retain their meanings from Equation 9. In particular, q indexes a probe mini-batch, not an individual example. We use the following condition when comparing the main losses of different candidates.

Assumption E.1 (Shared forward computation)**.** All candidates use the same probe mini-batches, network parameters, initial neuronal states, and forward computation. Only the temporal backward paths and the grouping of logits into segment losses can differ. Consequently,

$$\mathbf{z}_{q,t}^{(\mathcal{P})} = \mathbf{z}_{q,t}, \qquad \mathcal{P} \in \mathcal{C}, \quad q = 1, \ldots, Q, \quad t = 1, \ldots, T. \tag{11}$$

Here, $\mathbf{z}_{q,t}$ is the common forward output at time step t for probe mini-batch q. This condition requires the same random choices and normalization statistics across candidates. Detaching a backward path does not change its forward value; resetting a state or changing the temporal groups used by normalization can change that value. Thus, carrying neuronal states alone is not sufficient to establish this assumption.

E.1.2 MONITORED GRADIENTS AND THE TRAINING UPDATE.

We extend $\mathbf{g}_t$, $\mathbf{h}_t$, μ_t, and σ_t in Equation 1 by adding the probe index q and partition superscript $(\mathcal{P})$. For the full-update analysis, we assume that all trainable scalar parameters are monitored, so that $N_g = d$. The case $N_g < d$ is discussed in Subsection E.6. All candidates use the same ordering of gradient entries and the same loss normalization.

Assumption E.2 (Consistent temporal gradient aggregation)**.** The monitored gradients in each time step decompose the gradient accumulated before the single optimizer update described in the main text:

$$\mathbf{G}_q^{(\mathcal{P})} = \frac{1}{T} \sum_{t=1}^{T} \mathbf{g}_{q,t}^{(\mathcal{P})}. \tag{12}$$

Here, $\mathbf{G}_q^{(\mathcal{P})}$ is the accumulated gradient for probe mini-batch q, before momentum, preconditioning, clipping, or other optimizer transformations. Equation 12 is the specialization of Equation 2 to $w_t = 1/T$. It is a condition on the meaning and scaling of the monitored gradients, not an additional normalization of the implemented update. A compatible gradient decomposition for the length-weighted segment losses is given in Equation 39.

E.1.3 AVERAGES OVER TIME STEPS.

Define

$$\overline{\sigma}_q^{(\mathcal{P})} = \frac{1}{T}\sum_{t=1}^{T}\sigma_{q,t}^{(\mathcal{P})}, \qquad \overline{\mu}_q^{(\mathcal{P})} = \frac{1}{T}\sum_{t=1}^{T}\mu_{q,t}^{(\mathcal{P})}, \qquad \overline{\mathbf{h}}_q^{(\mathcal{P})} = \mathbf{G}_q^{(\mathcal{P})} - \overline{\mu}_q^{(\mathcal{P})}\mathbf{1}. \tag{13}$$

The scalars $\overline{\sigma}_q^{(\mathcal{P})}$ and $\overline{\mu}_q^{(\mathcal{P})}$ are the time averages of the gradient standard deviation and gradient mean. The vector $\mathbf{1} \in \mathbb{R}^d$ has every entry equal to one, and $\overline{\mathbf{h}}_q^{(\mathcal{P})}$ is the mean-centered part of the accumulated gradient. By Assumption E.2, it also equals $T^{-1}\sum_t \mathbf{h}_{q,t}^{(\mathcal{P})}$. Equation 8 can therefore be written as $\mathcal{L}_{\mathrm{std}}(\mathcal{P}) = Q^{-1}\sum_q \overline{\sigma}_q^{(\mathcal{P})}$. This is the original mean of standard deviations across gradient entries, not a variance across time steps or a root-mean-square replacement.

E.2 HOW GRADIENT DISPERSION BOUNDS TEMPORAL CREDIT AGGREGATION

The gradient standard deviation gives upper bounds on the size of the mean-centered aggregated gradient and its sensitivity to temporal alignment.

Proposition E.3 (Magnitude and alignment-sensitivity bounds)**.** *Under Assumption E.2, each probe mini-batch and candidate partition satisfy*

$$\left\|\overline{\mathbf{h}}_q^{(\mathcal{P})}\right\|_2 \le \sqrt{d}\,\overline{\sigma}_q^{(\mathcal{P})}, \qquad \left\|\mathbf{G}_q^{(\mathcal{P})}\right\|_2^2 \le d\left(\overline{\mu}_q^{(\mathcal{P})}\right)^2 + d\left(\overline{\sigma}_q^{(\mathcal{P})}\right)^2. \tag{14}$$

Furthermore, the sum of the alignment sensitivities in Equation 6 is

$$\sum_{1\le t<s\le T}\left|\frac{\partial\|\mathbf{G}_q^{(\mathcal{P})}\|_2^2}{\partial\rho_{q,ts}^{(\mathcal{P})}}\right| = d\left[\left(\overline{\sigma}_q^{(\mathcal{P})}\right)^2 - \frac{1}{T^2}\sum_{t=1}^{T}\left(\sigma_{q,t}^{(\mathcal{P})}\right)^2\right] \le d\left(\overline{\sigma}_q^{(\mathcal{P})}\right)^2. \tag{15}$$

Here, $\rho_{q,ts}^{(\mathcal{P})}$ is the normalized alignment of the mean-centered gradients at time steps t and s, with the same definition as in Equation 5. The derivative holds the means on the gradients in each time step and standard deviations fixed. When either standard deviation is zero, the corresponding interference coefficient and sensitivity are zero; its otherwise undefined normalized alignment can be assigned zero without changing the formula.

Proof. Equation 1 gives $\|\mathbf{h}_{q,t}^{(\mathcal{P})}\|_2 = \sqrt{d}\,\sigma_{q,t}^{(\mathcal{P})}$. The triangle inequality then gives the first bound in Equation 14. Since $\overline{\mathbf{h}}_q^{(\mathcal{P})}$ has zero coordinate sum, it is orthogonal to $\mathbf{1}$. This gives the second bound after separating the coordinate-mean component. For Equation 15, substitute $w_t = w_s = 1/T$ into Equation 6 and use $2\sum_{t<s}\sigma_{q,t}^{(\mathcal{P})}\sigma_{q,s}^{(\mathcal{P})} = (\sum_t \sigma_{q,t}^{(\mathcal{P})})^2 - \sum_t(\sigma_{q,t}^{(\mathcal{P})})^2$. □

E.2.1 A DIRECT BOUND USING THE IMPLEMENTED LOSS.

Suppose $\overline{\sigma}_q^{(\mathcal{P})} \le B_\sigma$ for every probe and candidate, where $B_\sigma > 0$ is one common bound for the entire comparison. Then $(\overline{\sigma}_q^{(\mathcal{P})})^2 \le B_\sigma\overline{\sigma}_q^{(\mathcal{P})}$ implies

$$\begin{aligned}\frac{1}{Q}\sum_{q=1}^{Q}\left\|\overline{\mathbf{h}}_q^{(\mathcal{P})}\right\|_2^2 &\le dB_\sigma\mathcal{L}_{\mathrm{std}}(\mathcal{P}),\\ \frac{1}{Q}\sum_{q=1}^{Q}\sum_{t<s}\left|\frac{\partial\|\mathbf{G}_q^{(\mathcal{P})}\|_2^2}{\partial\rho_{q,ts}^{(\mathcal{P})}}\right| &\le dB_\sigma\mathcal{L}_{\mathrm{std}}(\mathcal{P}).\end{aligned} \tag{16}$$

Thus, decreasing Equation 8 decreases these common upper bounds. The same B_σ must be used for all candidates; it must not be changed separately to improve a particular candidate's bound.

E.2.2 WHAT THESE BOUNDS DO NOT IMPLY.

The result bounds the effect of temporal alignment on the squared gradient magnitude; it is not a derivative with respect to a partition boundary. A lower upper bound does not require the realized gradient norm to be lower for every candidate. Also, the temporal concentration Γ_i in Equation 4 and the effective support in Equation 3 are scale-invariant. Reducing gradient scale can lower $\mathcal{L}_{\mathrm{std}}$ without changing either quantity. Consequently, the bounds above do not assume or establish that smaller dispersion alone balances all temporal contributions or guarantees better classification accuracy.

E.3 HOW THE MAIN LOSS MEASURES PREDICTION DISAGREEMENT

The main classification loss equals the full-sequence classification loss plus a nonnegative penalty for disagreement between segment and full-sequence predictions.

E.3.1 PREDICTIONS FOR EACH EXAMPLE.

Let b_q be the number of examples in probe mini-batch q, and let $n = 1, \ldots, b_q$ index those examples. The vectors $\mathbf{z}_{q,t,n} \in \mathbb{R}^C$ and $\mathbf{y}_{q,n} \in \mathbb{R}^C$ are the logits and one-hot label for example n, where C is the number of classes. They are the individual-example entries of the batched quantities $\mathbf{z}_{q,t}$ and $\mathbf{y}_q$ in the main text. For this analysis, the batched ℓ_{cls} is the mean Cross-Entropy over its examples. Under Assumption E.1, define

$$\begin{aligned} \overline{\mathbf{z}}_{q,m,n} &= \frac{1}{k_m} \sum_{t \in \mathcal{I}_m(\mathcal{P})} \mathbf{z}_{q,t,n}, \quad \mathbf{p}_{q,m,n} = \mathrm{softmax}(\overline{\mathbf{z}}_{q,m,n}), \\ \overline{\mathbf{z}}_{q,n} &= \frac{1}{T} \sum_{t=1}^{T} \mathbf{z}_{q,t,n}, \qquad\qquad \mathbf{p}_{q,n} = \mathrm{softmax}(\overline{\mathbf{z}}_{q,n}). \end{aligned} \tag{17}$$

The barred logit vectors average outputs within one segment or over the full sequence. The vectors $\mathbf{p}_{q,m,n}$ and $\mathbf{p}_{q,n}$ are the corresponding class probabilities. The quantities indexed by m depend on $\mathcal{P}$, which is omitted from their notation for readability. The full-sequence quantities do not depend on $\mathcal{P}$ during a candidate comparison. Their logit averages satisfy $\overline{\mathbf{z}}_{q,n} = \sum_m (k_m/T)\overline{\mathbf{z}}_{q,m,n}$.

E.3.2 FULL-SEQUENCE LOSS AND DISAGREEMENT.

We instantiate the reference training objective F from Equation 7 by

$$F(\boldsymbol{\theta}) = \frac{1}{Q} \sum_{q=1}^{Q} \frac{1}{b_q} \sum_{n=1}^{b_q} \ell_{\mathrm{cls}}(\overline{\mathbf{z}}_{q,n}, \mathbf{y}_{q,n}). \tag{18}$$

This is the full-sequence Cross-Entropy averaged first within each probe mini-batch and then over the Q probes. The dependence of the logits on $\boldsymbol{\theta}$ is implicit. For the algebraic results only its current value is needed; the differentiable reference used for the descent result is specified in Subsections E.5 and E.6. Define the disagreement quantity

$$\mathcal{R}(\mathcal{P}) = \frac{1}{Q} \sum_{q=1}^{Q} \frac{1}{b_q} \sum_{n=1}^{b_q} \sum_{m=1}^{M} \frac{k_m}{T} \mathrm{KL}(\mathbf{p}_{q,n} \| \mathbf{p}_{q,m,n}). \tag{19}$$

Here, $\mathrm{KL}(\mathbf{p}\|\mathbf{p}') = \sum_{c=1}^{C} p_c \log(p_c/p'_c)$ is the Kullback–Leibler divergence between two class-probability vectors, using natural logarithms. The index c denotes a class. Its first argument is the full-sequence prediction, not the segment prediction. The scalar $\mathcal{R}(\mathcal{P})$ is nonnegative and averages this disagreement with the same segment weights as Equation 9. It is introduced only to analyze the existing main loss, not as a new training term.

Proposition E.4 (Exact decomposition of the main loss)**.** *Under Assumption E.1, the Cross-Entropy main loss in Equation 9 satisfies*

$$\mathcal{L}_{\mathrm{main}}(\mathcal{P}) = F(\boldsymbol{\theta}) + \mathcal{R}(\mathcal{P}). \tag{20}$$

Proof. For a logit vector $\mathbf{v}$, write $\psi(\mathbf{v}) = \log \sum_{c=1}^{C} \exp(v_c)$. This log-sum-exp function gives $\ell_{\text{cls}}(\mathbf{v}, \mathbf{y}) = \psi(\mathbf{v}) - \mathbf{y}^\top \mathbf{v}$ for a one-hot label $\mathbf{y}$. Expanding the logarithms in the KL divergence gives

$$\text{KL}(\mathbf{p}_{q,n} \| \mathbf{p}_{q,m,n}) = \psi(\overline{\mathbf{z}}_{q,m,n}) - \psi(\overline{\mathbf{z}}_{q,n}) - \mathbf{p}_{q,n}^\top (\overline{\mathbf{z}}_{q,m,n} - \overline{\mathbf{z}}_{q,n}). \tag{21}$$

Average this identity over segments with weights k_m/T. The final linear term vanishes because the weighted segment logit average equals the full-sequence logit average. The label-dependent linear terms in the corresponding difference of Cross-Entropies vanish for the same reason. Averaging over examples and probes proves Equation 20. □

Interpretation. At a fixed parameter snapshot, the only partition-dependent part of the main loss is $\mathcal{R}(\mathcal{P})$. It is zero exactly when every segment predicts the same class probabilities as the full sequence for each example. Those common predictions need not be correct. Thus, the main-loss term measures prediction consistency during candidate selection; it does not by itself guarantee a useful or nonzero truncated gradient.

E.4 What an accepted partition guarantees

An accepted split reduces the measured gradient dispersion, whereas an accepted merge can trade higher dispersion for lower prediction disagreement.

E.4.1 The objective used for candidate comparison.

Substituting Equation 20 into Equation 10 gives

$$\mathcal{J}(\mathcal{P}) = \lambda_{\text{main}} F(\boldsymbol{\theta}) + \mathcal{L}_{\text{std}}(\mathcal{P}) + \lambda_{\text{main}} \mathcal{R}(\mathcal{P}). \tag{22}$$

The first term is common to all candidates. Therefore, the original objective compares gradient dispersion and prediction disagreement without modifying either loss or their scaling factor λ_{main}. We take $\lambda_{\text{main}} > 0$ in the following results.

Corollary E.5 (Dispersion change after accepting a candidate)**.** *Suppose Assumption E.1 holds. Let $\mathcal{P}$ be the current partition and $\mathcal{P}'$ an accepted candidate with*

$$\mathcal{J}(\mathcal{P}') \le \mathcal{J}(\mathcal{P}) - \varepsilon_{\text{sel}}, \qquad \varepsilon_{\text{sel}} > 0. \tag{23}$$

Then

$$\mathcal{L}_{\text{std}}(\mathcal{P}') - \mathcal{L}_{\text{std}}(\mathcal{P}) \le -\varepsilon_{\text{sel}} - \lambda_{\text{main}} \big[\mathcal{R}(\mathcal{P}') - \mathcal{R}(\mathcal{P})\big]. \tag{24}$$

If $\mathcal{P}'$ is obtained only by splitting segments of $\mathcal{P}$, then

$$\mathcal{R}(\mathcal{P}') \ge \mathcal{R}(\mathcal{P}), \qquad \mathcal{L}_{\text{std}}(\mathcal{P}') \le \mathcal{L}_{\text{std}}(\mathcal{P}) - \varepsilon_{\text{sel}}. \tag{25}$$

Here, ε_{sel} is a positive accepted improvement in the combined loss, consistent with the threshold-based acceptance rule in the main text. A partition obtained only by splitting is also called a refinement: each new segment lies within an old segment. The result concerns the measured losses on the shared probes at the current parameters.

Proof. Subtract Equation 22 for the two candidates and apply Equation 23. For a refinement, each original segment's mean logit vector is a length-weighted average of its new segments' mean logit vectors. Cross-Entropy is convex in the logits, so Jensen's inequality shows that splitting cannot decrease the length-weighted main loss. Equation 20 then gives $\mathcal{R}(\mathcal{P}') \ge \mathcal{R}(\mathcal{P})$. Substitution into Equation 24 proves Equation 25. □

E.4.2 Splitting and merging have different effects.

A split is accepted only when its dispersion reduction compensates for any increase in prediction disagreement. A merge can reduce disagreement and may therefore be accepted even if dispersion increases. Accordingly, the combined objective guarantees dispersion reduction for accepted refinements, not for every possible accepted change. The same distinction applies to a boundary shift that is not a refinement.

E.5 How the objective bounds the one-step training loss

Under explicit smoothness and gradient-discrepancy conditions, a lower partition selection objective gives a lower upper bound on the average one-step loss.

E.5.1 Separating segment supervision from temporal truncation.

Let $F_{\mathcal{P}}$ denote the segment classification objective in Equation 9, but differentiated through all temporal paths. Its value at the current parameters is $\mathcal{L}_{\text{main}}(\mathcal{P})$. Its full temporal gradient is generally different from the accumulated truncated gradient. In this subsection, F and $F_{\mathcal{P}}$ are evaluated using one common differentiable model that matches the current forward values and the backward rules under analysis. For a hard-spike SNN, this is a local reference model rather than an assumed globally smooth hard-spike loss; a construction is given in Subsection E.6.

For each segment and example, let

$$\mathbf{J}_{q,m,n} = \frac{\partial \overline{\mathbf{z}}_{q,m,n}}{\partial \boldsymbol{\theta}} \in \mathbb{R}^{C\times d}, \qquad \|\mathbf{J}_{q,m,n}\|_{\text{op}} \le B_z. \tag{26}$$

The matrix $\mathbf{J}_{q,m,n}$ is the full temporal Jacobian of the segment-averaged logits. The operator norm $\|\cdot\|_{\text{op}}$ is the largest singular value, and B_z is a common bound for all candidates, segments, examples, and probes at the current parameters. It bounds how much the logit vector changes under a small parameter perturbation.

Lemma E.6 (Gradient change caused by segment supervision)**.** *Under Assumption E.1 and Equation 26,*

$$\|\nabla F_{\mathcal{P}}(\boldsymbol{\theta}) - \nabla F(\boldsymbol{\theta})\|_2 \le B_z\sqrt{2\mathcal{R}(\mathcal{P})}. \tag{27}$$

Here and below, ∇ denotes a gradient with respect to the network parameters $\boldsymbol{\theta}$. This bound measures the change caused by attaching classification losses to segments while retaining full temporal differentiation. It does not yet include temporal truncation.

Proof. The derivative of Cross-Entropy with respect to its logits is the predicted probability vector minus the label vector. Differentiating the weighted logit-average identity in Equation 17 therefore yields

$$\nabla F_{\mathcal{P}} - \nabla F = \frac{1}{Q}\sum_{q=1}^{Q}\frac{1}{b_q}\sum_{n=1}^{b_q}\sum_{m=1}^{M}\frac{k_m}{T}\mathbf{J}_{q,m,n}^{\top}(\mathbf{p}_{q,m,n} - \mathbf{p}_{q,n}). \tag{28}$$

All gradients in this equation are evaluated at the current $\boldsymbol{\theta}$. Pinsker's inequality gives $\|\mathbf{p}_{q,m,n} - \mathbf{p}_{q,n}\|_2 \le \|\mathbf{p}_{q,m,n} - \mathbf{p}_{q,n}\|_1 \le \sqrt{2\,\mathrm{KL}(\mathbf{p}_{q,n}\|\mathbf{p}_{q,m,n})}$. Apply the triangle inequality, the Jacobian bound, and the concavity of the square root to the weights $(1/Q)(1/b_q)(k_m/T)$, which sum to one. This proves Equation 27. □

E.5.2 The additional discrepancy caused by truncation.

Define

$$\overline{\mathbf{G}}^{(\mathcal{P})} = \frac{1}{Q}\sum_{q=1}^{Q}\mathbf{G}_q^{(\mathcal{P})}, \qquad \mathbf{e}_{\text{tr}}^{(\mathcal{P})} = \overline{\mathbf{G}}^{(\mathcal{P})} - \nabla F_{\mathcal{P}}(\boldsymbol{\theta}). \tag{29}$$

The vector $\overline{\mathbf{G}}^{(\mathcal{P})}$ averages the accumulated gradients across probes. The residual $\mathbf{e}_{\text{tr}}^{(\mathcal{P})}$ measures the difference between that average and full temporal differentiation of the same segment losses. This residual is not assumed to be zero. Controlling an error caused by temporal truncation is also central to the gradient-bias analysis of Aicher et al. (2020); the bound below keeps this error separate from prediction disagreement.

Assumption E.7 (Local smoothness and common bounds)**.** The common reference objective F is L-smooth on a region containing every update line segment considered below, so Equation 7 applies. The Jacobian condition in Equation 26 holds. There are constants $B_\sigma > 0$, $B_\mu \ge 0$, and $B_{\text{tr}} \ge 0$, shared by all $\mathcal{P} \in \mathcal{C}$, such that

$$\overline{\sigma}_q^{(\mathcal{P})} \le B_\sigma \quad \text{for every } q, \qquad \frac{1}{Q}\sum_{q=1}^{Q}(\overline{\mu}_q^{(\mathcal{P})})^2 \le B_\mu^2, \qquad \|\mathbf{e}_{\text{tr}}^{(\mathcal{P})}\|_2 \le B_{\text{tr}}. \tag{30}$$

The constant L is the same smoothness constant used in Equation 7. The constant B_σ is the dispersion bound used in Equation 16. The constant B_μ bounds the root-mean-square coordinate mean of the accumulated gradients, and B_{tr} bounds the truncation discrepancy. These constants are held fixed while candidates are compared; the partition search is not claimed to estimate or enforce them.

Theorem E.8 (One-step loss bound for the partition selection objective)**.** *Suppose Assumptions E.1, E.2, and E.7 hold, with $N_g = d$ and $\lambda_{\mathrm{main}} > 0$. For a learning rate $\eta > 0$, define*

$$\boldsymbol{\theta}_{q,+}^{(\mathcal{P})} = \boldsymbol{\theta} - \eta \mathbf{G}_q^{(\mathcal{P})}, \qquad \kappa = \max\left\{ \frac{L\eta^2 d B_\sigma}{2}, \frac{2\eta B_z^2}{\lambda_{\mathrm{main}}} \right\}. \tag{31}$$

Then

$$\begin{aligned} &\frac{1}{Q}\sum_{q=1}^{Q} F(\boldsymbol{\theta}_{q,+}^{(\mathcal{P})}) \le \mathcal{U}(\mathcal{P}), \\ &\mathcal{U}(\mathcal{P}) = F(\boldsymbol{\theta}) - \frac{\eta}{2}\|\nabla F(\boldsymbol{\theta})\|_2^2 + \eta B_{\mathrm{tr}}^2 + \frac{L\eta^2 d}{2} B_\mu^2 \\ &\qquad + \kappa\big[\mathcal{L}_{\mathrm{std}}(\mathcal{P}) + \lambda_{\mathrm{main}}\mathcal{R}(\mathcal{P})\big]. \end{aligned} \tag{32}$$

Moreover, accepting a candidate as in Equation 23 gives

$$\mathcal{U}(\mathcal{P}') \le \mathcal{U}(\mathcal{P}) - \kappa\varepsilon_{\mathrm{sel}}. \tag{33}$$

The vector $\boldsymbol{\theta}_{q,+}^{(\mathcal{P})}$ is a hypothetical one-step SGD update from the same current parameters using probe q. The average in Equation 32 therefore concerns separate one-step updates, not Q consecutive training updates. The scalar κ is a common coefficient that combines the two loss-dependent bounds. The quantity $\mathcal{U}(\mathcal{P})$ is an upper bound on the average post-update reference loss, not an extra objective evaluated by LT-BPTT.

Proof. Write $\mathbf{f} = \nabla F(\boldsymbol{\theta})$ for the reference gradient and $\mathbf{e}^{(\mathcal{P})} = \overline{\mathbf{G}}^{(\mathcal{P})} - \mathbf{f}$ for its total discrepancy from the averaged truncated gradient. Lemma E.6 and Equation 29 give

$$\|\mathbf{e}^{(\mathcal{P})}\|_2^2 \le 2\|\mathbf{e}_{\mathrm{tr}}^{(\mathcal{P})}\|_2^2 + 2\|\nabla F_{\mathcal{P}} - \nabla F\|_2^2 \le 2B_{\mathrm{tr}}^2 + 4B_z^2\mathcal{R}(\mathcal{P}). \tag{34}$$

Using $\langle \mathbf{f}, \mathbf{e}^{(\mathcal{P})}\rangle \ge -\frac{1}{2}\|\mathbf{f}\|_2^2 - \frac{1}{2}\|\mathbf{e}^{(\mathcal{P})}\|_2^2$ therefore yields

$$\langle \mathbf{f}, \overline{\mathbf{G}}^{(\mathcal{P})}\rangle \ge \frac{1}{2}\|\mathbf{f}\|_2^2 - B_{\mathrm{tr}}^2 - 2B_z^2\mathcal{R}(\mathcal{P}). \tag{35}$$

The left-hand side is the descent-alignment term in Equation 7, averaged over probes. Separately, Proposition E.3 and the common bounds imply

$$\frac{1}{Q}\sum_{q=1}^{Q}\|\mathbf{G}_q^{(\mathcal{P})}\|_2^2 \le dB_\mu^2 + dB_\sigma\mathcal{L}_{\mathrm{std}}(\mathcal{P}). \tag{36}$$

This is the squared-gradient term in the same descent inequality, averaged over probes. Apply Equation 7 to each update and average:

$$\begin{aligned} \frac{1}{Q}\sum_{q=1}^{Q} F(\boldsymbol{\theta} - \eta\mathbf{G}_q^{(\mathcal{P})}) &\le F(\boldsymbol{\theta}) - \eta\langle \mathbf{f}, \overline{\mathbf{G}}^{(\mathcal{P})}\rangle + \frac{L\eta^2}{2Q}\sum_{q=1}^{Q}\|\mathbf{G}_q^{(\mathcal{P})}\|_2^2 \\ &\le F(\boldsymbol{\theta}) - \frac{\eta}{2}\|\mathbf{f}\|_2^2 + \eta B_{\mathrm{tr}}^2 + \frac{L\eta^2 d}{2}B_\mu^2 \\ &\quad + 2\eta B_z^2\mathcal{R}(\mathcal{P}) + \frac{L\eta^2 d B_\sigma}{2}\mathcal{L}_{\mathrm{std}}(\mathcal{P}). \end{aligned} \tag{37}$$

Both loss-dependent quantities are nonnegative. The definition of κ bounds the last line by $\kappa[\mathcal{L}_{\mathrm{std}}(\mathcal{P}) + \lambda_{\mathrm{main}}\mathcal{R}(\mathcal{P})]$. This proves Equation 32. Finally, Equation 22 shows that the bracket equals $\mathcal{J}(\mathcal{P}) - \lambda_{\mathrm{main}}F(\boldsymbol{\theta})$. All remaining terms are common to the candidates, which proves Equation 33. □

E.5.3 WHEN THE REFERENCE LOSS IS GUARANTEED TO DECREASE.

A sufficient condition for the average one-step reference loss to be strictly smaller than its current value is

$$\frac{\eta}{2}\|\nabla F(\boldsymbol{\theta})\|_2^2 > \eta B_{\mathrm{tr}}^2 + \frac{L\eta^2 d}{2} B_\mu^2 + \kappa\big[\mathcal{L}_{\mathrm{std}}(\mathcal{P}) + \lambda_{\mathrm{main}}\mathcal{R}(\mathcal{P})\big]. \tag{38}$$

The left-hand side represents the available descent from the reference gradient. The right-hand side bounds the penalties from truncation discrepancy, the coordinate-mean component, gradient dispersion, and prediction disagreement. Reducing the partition selection objective reduces the last bracket while the common constants remain fixed. This makes the sufficient condition easier to satisfy, but the existence of finite bounds alone does not ensure that it holds. In particular, a smaller upper bound does not establish that two candidates' realized post-update losses have the same ordering.

E.6 SCOPE FOR SNN TRAINING AND IMPLEMENTATION

The guarantees concern the monitored gradients and a local reference loss, not automatically classification accuracy or the updates of every optimizer.

E.6.1 A GRADIENT DECOMPOSITION CONSISTENT WITH THE SEGMENT LOSS.

The entries in Equation 8 must be tied to the accumulated training gradient to apply the full-update bounds. One choice satisfying Assumption E.2 is

$$\mathbf{g}_{q,t}^{(\mathcal{P})} = \frac{1}{b_q}\sum_{n=1}^{b_q}\left(\mathbf{J}_{q,t,n}^{(\mathcal{P})}\right)^\top (\mathbf{p}_{q,m,n} - \mathbf{y}_{q,n}), \qquad t \in \mathcal{I}_m(\mathcal{P}). \tag{39}$$

Here, $\mathbf{J}_{q,t,n}^{(\mathcal{P})}$ is the Jacobian of time step's logit vector $\mathbf{z}_{q,t,n}$ with the incoming state of its segment held constant in differentiation. It includes all retained backward paths within that segment. The vector $\mathbf{p}_{q,m,n} - \mathbf{y}_{q,n}$ is the logit derivative of the segment-level Cross-Entropy. For each output time step, the segment weight k_m/T cancels the factor $1/k_m$ from logit averaging, leaving $1/T$. Summing Equation 39 therefore gives the accumulated length-weighted segment gradient in Equation 12. Other decompositions are admissible if they satisfy the same identity. Computing an independent Cross-Entropy from each time step's logits generally gives a different gradient decomposition. Monitoring and accumulation must therefore use the same loss and normalization.

E.6.2 WHY TRUNCATION DISCREPANCY REMAINS A SEPARATE CONDITION.

For exact differentiation of a recurrent reference model with a parameter-independent initial state, the chain rule identifies the removed boundary contributions:

$$\mathbf{e}_{\mathrm{tr}}^{(\mathcal{P})} = -\frac{1}{Q}\sum_{q=1}^{Q}\sum_{m=2}^{M}\frac{k_m}{T}\left(\frac{\partial \mathbf{s}_{q,m}^{\mathrm{in}}}{\partial \boldsymbol{\theta}}\right)^\top \nabla_{\mathbf{s}_{q,m}^{\mathrm{in}}}\ell_{\mathrm{cls}}(\overline{\mathbf{z}}_{q,m}, \mathbf{y}_q). \tag{40}$$

The vector $\mathbf{s}_{q,m}^{\mathrm{in}}$ collects the neuronal states entering segment m for all examples in probe q. Its full Jacobian describes how earlier parameter uses affect that incoming state. The final gradient describes how the segment loss depends on the incoming state, with the parameters within the segment held fixed. The batched quantity $\overline{\mathbf{z}}_{q,m}$ collects the example-level averages in Equation 17, and its classification loss is averaged over those examples. Truncating the boundary removes the product of these two derivatives. Neither $\mathcal{L}_{\mathrm{std}}$ nor $\mathcal{R}$ alone bounds all of these products, which is why B_{tr} remains explicit in Theorem E.8.

E.6.3 HARD SPIKES AND A LOCAL DIFFERENTIABLE REFERENCE.

The magnitude bounds, the main-loss decomposition, and the accepted-split result do not require a differentiable spike function. The one-step loss bound does require a smooth reference objective. Surrogate gradients are generally not gradients of a globally defined surrogate loss (Gygax & Zenke, 2025); a smooth backward derivative alone does not make the hard-spike loss smooth. A simple

local reference can be constructed for each spike occurrence by fixing its current threshold-centered membrane value v_0 and replacing its spike function by

$$s_{\mathrm{loc}}(v; v_0) = \mathbb{I}\{v_0 \geq 0\} + \gamma_0(v - v_0). \tag{41}$$

The variable v is the membrane value measured relative to the firing threshold, $\mathbb{I}$ is the indicator function, and γ_0 is the prescribed surrogate derivative evaluated at v_0. Both v_0 and γ_0 are frozen for the local comparison. At $v = v_0$, this affine function matches the hard forward spike and the prescribed backward derivative. If the remaining operations and their backward rules admit the same differentiable reference, applying this construction to the unfolded network matches its current logits and backward Jacobians. Local smoothness must still hold over the update region. Theorem E.8 then bounds this reference loss, not the hard-spike loss after arbitrary threshold crossings. Rebuilding the reference at a new parameter snapshot does not imply a single global surrogate loss.

E.6.4 Monitored coordinates and optimizer transformations.

When $N_g < d$, the aggregation bounds hold in the monitored coordinates with d replaced by N_g. A bound for the full parameter update additionally requires control of the unmonitored coordinates. The coordinate-mean component is also not controlled by a centered standard deviation, which explains the separate B_μ term. Finally, Theorem E.8 uses the SGD-form update in Equation 7. Table 4 also includes Adam, so the theorem should not be read as an optimizer-specific guarantee for every reported configuration. Momentum, adaptive preconditioning, weight decay, and clipping require analysis of the transformed update and any optimizer state; the raw-gradient aggregation results remain valid before those transformations.

E.6.5 What the analysis establishes.

The existing objective controls two different quantities: $\mathcal{L}_{\mathrm{std}}$ bounds mean-centered temporal credit aggregation, while the partition-dependent part of $\mathcal{L}_{\mathrm{main}}$ bounds the gradient change caused by segment supervision. Their sum gives the conditional one-step bound above, and accepted refinements also reduce the measured dispersion directly. These statements apply to the evaluated probes at the current parameter snapshot. They do not establish a population guarantee from a finite probe set, monotonic loss reduction throughout the next epoch, or an unconditional increase in test accuracy.