DNASNET: DECOUPLED NEURAL ARCHITECTURE SEARCH DRIVEN BY TASK-CONDITIONED SPIKE TIMING

Anonymous authors
Paper under double-blind review

ABSTRACT

Spiking neural networks (SNNs) promise energy-efficient computation, but differentiable search over temporally unrolled SNNs is costly. DNASNet decouples weight-sharing NAS: spatio-temporal backpropagation (STBP) learns shared weights, while an STDP-inspired rule updates architectures from task-induced spike timing without architecture-level backpropagation. With network state and timing evidence fixed, the STDP utility is the gradient of an explicit concave temporal-balance potential, and the normalized update equals entropy mirror ascent on node-wise probability simplices. DNASNet produces stable architectures across random runs, achieves near-state-of-the-art performance on CIFAR-10, CIFAR-100, and DVS-Gesture, and yields competitive results on DVS-CIFAR-10 and ImageNet-1K. Compared with bilevel optimization, it reduces peak search memory by up to 22% and search time by up to 51%. Ablations further confirm the importance of timing information for architectural consistency and retraining performance.

1 INTRODUCTION

Biologically inspired SNNs (Maass, 1997) exploit sparse events for potentially energy-efficient neuromorphic computation (Roy et al., 2019; Davies et al., 2018). Deep SNN architectures have progressed from adapted VGG and residual networks (Sengupta et al., 2019) to spike-specific residual connections and attention (Fang et al., 2021a; Zhou et al., 2023; Yao et al., 2023). However, ANN-derived connectivity need not be optimal for temporal processing or spike efficiency (Kim et al., 2022; Na et al., 2022).

To address this design challenge, SNN-specific neural architecture search (NAS) methods explore training-free scoring (Kim et al., 2022; Wang et al., 2024), evolutionary selection (Na et al., 2022), and differentiable optimization, as in SpikeDHS and MA-DARTS (Che et al., 2022; Man et al., 2024). Differentiable SNN search couples architecture optimization with spatio-temporal backpropagation (STBP) (Wu et al., 2018), whose temporally unrolled computation incurs substantial memory and runtime costs (Meng et al., 2023). This raises a question: *Can SNN architecture search be driven by task-conditioned spike timing without architecture-level backpropagation?*

We propose DNASNet to address this question by decoupling architecture and weight updates: local timing evidence guides architecture selection, while STBP trains shared weights. Our architecture update is inspired by spike-timing-dependent plasticity (STDP), which changes synapses according to local pre/post spike timing (Bi & Poo, 1998; Caporale & Dan, 2008). Competition and normalization support self-organized connectivity (Gütig et al., 2003; Lazar et al., 2009); NeuEvo already combines local circuit plasticity with global-error weight learning (Shen et al., 2023), but still solves a two-level problem. With local weights w_l and global weights w_g, they optimize $\min_{w_l} \mathcal{L}_{w_l}(\arg\min_{w_g} \mathcal{L}_w(w_g w_l), w_l)$ by alternating updates with a one-step lookahead approximation, using $\nabla \mathcal{L}_{w_l}(w_g - \xi \nabla \mathcal{L}_w(w_g w_l), w_l)$. DNASNet instead uses task-conditioned spike timing for forward-only architecture updates. Each iteration constructs supervised lookahead weights, freezes the network state and routing, and records source–candidate spike co-activity in a forward pass. STDP evidence updates node-wise architecture logits, and the resulting routing guides weight learning. Supervision thus conditions architecture selection through the trained network state, without an architecture backward graph. Node-wise routing follows entropy mirror ascent on a simplex, with

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.

STDP utilities given by the gradient of an explicit concave quadratic potential. This conditional potential varies with network state and sampled evidence as supervised learning updates network weights.

Our main contributions are as follows:

- We propose DNASNet, an architecture search method that decouples shared-weight training from forward-only architecture updates. Relative to a differentiable SNN search baseline, it reduces peak search memory by up to 22% and search time by up to 51%.
- We derive an explicit concave quadratic potential and an exact entropy mirror recursion from a probability-dependent STDP rule. Under explicit assumptions, we analyze routing stability, training convergence, and online regret to characterize the update rule, its competitive behavior, and the analysis scope.
- We evaluate image benchmarks, search-rule ablations, and cross-modal transfer. DNASNet improves accuracy over NeuEvo by 0.08, 0.85, and 4.53 percentage points on CIFAR-10, CIFAR-100, and ImageNet-1K, respectively.

2 Related Work

Network design and architecture search for spiking dynamics. STBP and surrogate gradients enable direct SNN training (Wu et al., 2018; Neftci et al., 2019); SEW-ResNet, Spikformer, SpikingResformer, and scalable spike-driven Transformers advance residual propagation and spiking attention (Fang et al., 2021a; Zhou et al., 2023; Shi et al., 2024; Yao et al., 2025). SLTT reduces temporal-backpropagation costs (Meng et al., 2023), whereas DNASNet targets architecture-update costs. For SNN-NAS, SNASNet scores initial spike-activation diversity and searches feedback connections (Kim et al., 2022); LightSNN performs training-free pruning with a sparsity-aware score (Abdennadher et al., 2025). AutoSNN combines a trained supernetwork with evolutionary search and spike-count fitness (Na et al., 2022); ES-NAS uses a branchless supernetwork and SynOps-aware optimization (Yan et al., 2024). Differentiable methods include hierarchical SpikeDHS (Che et al., 2022), attention-based MA-DARTS (Man et al., 2024), Spatial-Temporal Search over temporal parameters and spiking/non-spiking units (Che et al., 2024b). Auto-Spikformer jointly searches Transformer architectures and SNN parameters, AutoST uses training-free proxies, and Auto-FSDformer searches fully spike-driven Transformers (Che et al., 2024a; Wang et al., 2024; Yang & Liu, 2026). DNASNet's distinguishing feature is its task-conditioned timing signal during shared-weight training; feedback and temporal search already have precedents.

Local plasticity and data-conditioned structural learning. Weight dependence and competition shape STDP connectivity (Bi & Poo, 1998; Gütig et al., 2003). For SNN weight learning, task information enters through reward-modulated STDP in convolutional networks (Mozafari et al., 2019) or global error signals coupled with STDP-derived updates in SSTDP (Liu et al., 2021). Hybrid plasticity further integrates global supervised learning with meta-learned local plasticity (Wu et al., 2022). EIHL adapts network connectivity and switches between STDP and STBP according to network sparsity (Jiang et al., 2024). DNASNet assigns its STDP-inspired updates to architecture logits governing node-wise competition among candidate operations, with timing evidence collected under frozen supervised lookahead weights, while STBP optimizes the shared weights. MetaD2A instead learns dataset–architecture relationships from previously evaluated pairs (Lee et al., 2021).

Simplex geometry and optimization interpretations of STDP. Exponentiated gradient and entropy mirror methods optimize normalized nonnegative variables (Kivinen & Warmuth, 1997; Beck & Teboulle, 2003). GAEA applies this geometry to weight-sharing NAS with task-loss architecture gradients (Li et al., 2021). Dexheimer et al. (2025) relate a specified Hebbian rule to noisy gradient descent for a nonconvex loss on a probability simplex. Using these established tools, DNASNet characterizes its probability-dependent timing rule: with timing evidence frozen, the utility is the gradient of an explicit concave quadratic potential, and the normalized update is entropy mirror ascent. Our analysis concerns this conditional potential during task-conditioned shared-weight training.

3 PROPOSED METHOD: DNASNET

We define the search space, STDP utility and conditional potential, then the alternating weight and architecture updates.

3.1 SPIKING NEURONS

We use computationally efficient, biologically motivated leaky integrate-and-fire (LIF) neurons with discrete layer-wise dynamics at time t:

$$u_t^i = \left(1 - \frac{1}{\tau}\right) u_{t-1}^i + \frac{1}{\tau} X_t^i, \qquad S_t^i = g(u_t^i) = \begin{cases} 1, & u_t^i \geq V_{th} \\ 0, & u_t^i < V_{th} \end{cases}, \qquad X_t^i = W^{i-1} S_t^{i-1} \tag{1}$$

where u_t^i is the membrane potential in layer i, X_t^i is the input current, τ is the membrane time constant, and V_{th} is the firing threshold. A spike triggers a reset of the membrane potential. STBP uses a surrogate gradient for the spike function (Neftci et al., 2019). The trainable networks use parametric LIF (PLIF) neurons (Fang et al., 2021b), with $1/\tau = 1/(1 + \exp(-\beta))$.

3.2 SEARCH SPACE AND NODE-WISE ROUTING

As in DARTS (Liu et al., 2019), DNASNet constructs networks by stacking cells. Each searchable normal cell contains two input nodes, $0, 1$, and three intermediate nodes, $2, 3, 4$ (Figure 1(a)). Feedforward edges originate from lower-indexed nodes, giving the three intermediate nodes 2, 3, and 4 candidate incoming edges, respectively. Following SNASNet's findings on the benefits of feedback across time steps (Kim et al., 2022), we jointly search feedforward and feedback operations to capture dependencies in evolving spiking states. Node 3 can feed back to nodes $0, 1, 2$, and node 4 to nodes $0, 1, 2, 3$. Feedback returns outputs to preceding nodes with a one-step delay. Each operation's architecture parameter is its routing probability p; w denotes all operation and network parameters learned through backpropagation. Let $v \in \mathcal{V}$ index the *node-wise routing groups*: feedforward candidates are grouped by destination node, whereas feedback candidates are grouped by source node, with separate normalization for each group. Given an edge set $\mathcal{E}_v$ and an operation set $\mathcal{O}$, the candidate index set is $\mathcal{I}_v = \{(e, o) : e \in \mathcal{E}_v,\ o \in \mathcal{O}\}$, with cardinality $d_v = |\mathcal{I}_v|$. Inspired by competition in models of synaptic plasticity (Gütig et al., 2003; Lazar et al., 2009), *all* edge–operation candidates in each group jointly compete for routing probability mass, selecting input sources and operations in feedforward groups and feedback destinations and operations in feedback groups.

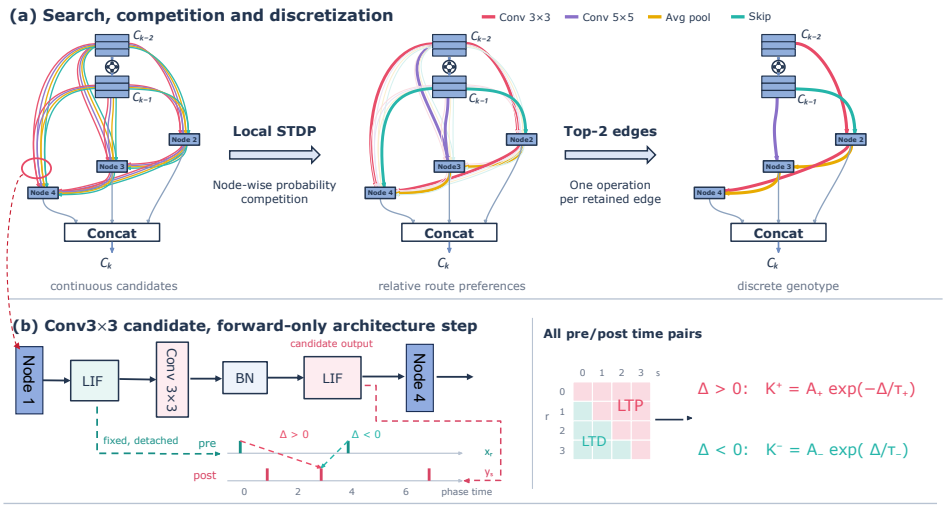


Figure 1: **DNASNet search and discretization.** STBP trains shared weights; local timing statistics update architecture logits. Discretization retains two predecessor edges per intermediate node and one operation per edge. Feedback (not shown) retains one edge per source node with one operation.

The candidate sets $\{\mathcal{I}_v\}_{v\in\mathcal{V}}$ define the supernetwork search space. For all image datasets, $\mathcal{O}$ contains {`3x3/5x5 convolution`, `3x3 average pool`, `skip connect`}, each producing observable output spike trains. The group logits $z_v \in \mathbb{R}^{d_v}$ define jointly normalized routing probabilities $p_v = \mathrm{softmax}(z_v) \in \mathrm{int}(\Delta^{d_v-1})$. The full architecture space is the product of these simplices, $\mathcal{P} = \prod_{v\in\mathcal{V}} \Delta^{d_v-1}$. There are nine feedforward edges and seven feedback edges, yielding approximately 1.42×10^7 architectures in the discrete search space. During search, p_v mixes candidate outputs; after search, candidates are ranked by their routing probabilities to obtain a discrete genotype. Appendix C.1 details the search-space design.

3.3 From Timing Evidence to Node-Level Competition

Node-wise routing defines candidate competition but requires a signal for probability updates. Each candidate's input–output spike timing is locally observable through forward activity, motivating an STDP-inspired rule that converts temporal relationships into signed routing evidence without backpropagating task loss through architecture parameters. In pair-based STDP, a presynaptic spike preceding a postsynaptic spike induces long-term potentiation (LTP), whereas the reverse order induces long-term depression (LTD). Both effects decay exponentially with the time interval. A representative multiplicative rule in this weight-dependent family (van Rossum et al., 2000; Gütig et al., 2003) is

$$g^{(k+1)} = g^{(k)} + \Delta g = g^{(k)} + \begin{cases} A_+ e^{-(t_{\mathrm{post}}-t_{\mathrm{pre}})/\tau_+}(g_{\max} - g^{(k)}), & t_{\mathrm{post}} - t_{\mathrm{pre}} > 0, \quad \text{(LTP)} \\ -A_- e^{(t_{\mathrm{post}}-t_{\mathrm{pre}})/\tau_-} g^{(k)}, & t_{\mathrm{post}} - t_{\mathrm{pre}} < 0, \quad \text{(LTD)} \end{cases} \tag{2}$$

where $g^{(k)}$ is the connection strength at update k, t_{pre} and t_{post} are spike times, $g_{\max}$ sets the saturation level for potentiation, and $A_\pm$ are the amplitude coefficients for potentiation and depression. We adapt potentiation–depression to architecture routing: timing evidence supplies the utility for updating logits z_v, while STBP trains the shared weights w. Supervised weight lookahead conditions this evidence on the task (Section 3.4).

Figure 1(b) pairs source spikes with each candidate's output spikes. Asymmetric exponential kernels aggregate LTP/LTD evidence. At architecture iteration t, a forward pass with network state w and routing frozen produces nonnegative evidence vectors $\widehat{\boldsymbol{A}}^+_{t,v}(w), \widehat{\boldsymbol{A}}^-_{t,v}(w) \in \mathbb{R}^{d_v}_+$ for each routing group. For potentiation scale $c_v > 0$ and $p_{t,v} \in \Delta^{d_v-1}$, the STDP utility is

$$\widehat{g}_{t,v}(p_{t,v} \mid w) = \widehat{\boldsymbol{A}}^+_{t,v}(w) \odot (c_v \mathbf{1} - p_{t,v}) - \widehat{\boldsymbol{A}}^-_{t,v}(w) \odot p_{t,v}. \tag{3}$$

Here, $\odot$ denotes elementwise multiplication, and $\widehat{g}_{t,v}$ is a signed score. We update the logits with architecture step size η_t and normalize:

$$z_{t+1,v} = z_{t,v} + \eta_t \widehat{g}_{t,v}, \qquad p_{t+1,v} = \mathrm{softmax}(z_{t+1,v}). \tag{4}$$

For $\eta_t > 0$, the probability ratio of a higher-utility candidate to a lower-utility candidate increases. The following proposition gives a geometric interpretation of this competition.

Proposition 1 (Exact entropy mirror interpretation)**.** *For each node-wise routing group v, Equation 4 is exactly equivalent to*

$$p_{t+1,v} = \operatorname*{arg\,max}_{q_v \in \Delta^{d_v-1}} \left\{ \eta_t \langle q_v, \widehat{g}_{t,v} \rangle - D_{\mathrm{KL}}(q_v \| p_{t,v}) \right\}. \tag{5}$$

Proposition 2 (Explicit conditional STDP potential and stochastic gradient decomposition)**.** *For any frozen network state w, define the per-iteration architecture potential as*

$$J_t(p \mid w) = C_t(w) - \frac{1}{2} \sum_{v\in\mathcal{V}} h_{t,v}^\top \big[(p_v - r_{t,v}) \odot (p_v - r_{t,v}) \big], \tag{6}$$

where $C_t(w)$ is independent of p, $h_{t,v} = \widehat{\boldsymbol{A}}^+_{t,v}(w) + \widehat{\boldsymbol{A}}^-_{t,v}(w)$, $r_{t,v} = c_v \widehat{\boldsymbol{A}}^+_{t,v}(w) \oslash h_{t,v}$, $\oslash$ denotes elementwise division. For a fixed network state, $J_t(\cdot \mid w)$ is concave and $\widehat{g}_t = \nabla_p J_t(p_t \mid w)$. For a random architecture minibatch, let $\mathcal{F}_t$ contain w, p_t, and all information available before that minibatch is sampled. Then

$$\widehat{g}_t = \nabla_p \overline{J}_t(p_t \mid w) + \xi_t, \qquad \mathbb{E}[\xi_t \mid \mathcal{F}_t] = 0. \tag{7}$$

where $\overline{J}_t(p \mid w) := \mathbb{E}[J_t(p \mid w) \mid \mathcal{F}_t]$ is obtained by replacing the observed evidence vectors with their conditional expectations, and ξ_t is centered sampling noise.

Both feedforward and feedback groups follow this entropy mirror update. Under a fixed network state, Proposition 2 gives each candidate an adaptive local LTP/LTD balance target $r_{t,v}$, determined by its LTP fraction and weighted by its total timing evidence $h_{t,v}$. The target applies wherever $h_{t,v} > 0$; we set $r_{t,v,i} = 0$ when $h_{t,v,i} = 0$. The simplex constraint couples candidates through competition, and the conditional potential characterizes their local temporal preferences. Appendix A gives the complete timing-statistic definitions and proposition proofs.

Empirical evaluation of the temporal-balance potential. Using the evidence measured in the architecture step, we define $E_t(p) := C_t - J_t(p \mid \widehat{w}_t)$, $H_{t,v} := \sum_i h_{t,v,i}$, and the normalized deviation $D_{t,v} := \sqrt{2E_{t,v}(p_{t,v})/H_{t,v}}$, with the latter defined only for groups where $H_{t,v} \neq 0$. For fixed evidence, energy decreases exactly when potential increases. The $h_{t,v}$-weighted root-mean-square deviation $D_{t,v}$ separates evidence-scale changes from routing movement toward the balance target. Figure 2 shows an overall decrease in energy (left) and normalized deviation (right) during search on DVS-Gesture, demonstrating that the routing probabilities are indeed attracted toward the balance targets.

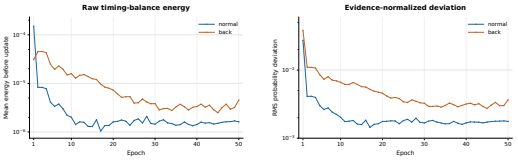


Figure 2: **Temporal balance.**

3.4 Decoupled Task Conditioning and Search Dynamics

For c-class classification, let $(x_i, y_i) \in \mathcal{D}$ with one-hot labels $y_i \in \{0,1\}^c$. We split the search data equally into disjoint sets $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{arch}}$. Denote the SNN classifier by $f(x_i; w, p)$. The per-example loss is $L_i^{\text{train}}(w,p) = \ell(f(x_i; w, p), y_i)$, and the supervised objective is $F(w,p) = \frac{1}{N}\sum_{i=1}^{N} L_i^{\text{train}}(w,p)$. DNASNet updates w using STBP and $p \in \mathcal{P}$ using timing utilities measured at a supervised lookahead state. DARTS solves the bilevel problem $\min_\alpha \mathcal{L}_{val}(w^*(\alpha), \alpha)$ subject to $w^*(\alpha) = \arg\min_w \mathcal{L}_{train}(w, \alpha)$ (Liu et al., 2019). DNASNet instead updates routing from timing evidence; Section 4 assesses architecture quality through validation-based selection and independent retraining. A complete training iteration consists of three sequential steps:

Step 1: Intermediate Weight Approximation ($\mathcal{D}_{train}$). At (w_t, p_t), draw a minibatch of n examples (x_i, y_i) from $\mathcal{D}_{\text{train}}$ and update the classifier to decrease their mean training loss:

$$\widehat{w}_t = w_t - \frac{\alpha_t}{n}\sum_{i=1}^{n} \nabla_w L_i^{\text{train}}(w_t, p_t). \tag{8}$$

Step 2: Gradient-Free Architecture Update ($\mathcal{D}_{arch}$). Freeze $(\widehat{w}_t, p_t)$ and run a forward pass on an independent minibatch drawn from $\mathcal{D}_{\text{arch}}$. Record spike statistics at source nodes and candidate outputs and detach them from gradient computation; neither labels nor a backward computational graph are required. Compute the utility using Equation 3, and update each node-wise group as

$$p_{t+1,v} = \text{softmax}(z_{t+1,v}) = \text{softmax}(z_{t,v} + \eta_t \widehat{g}_{t,v}), \tag{9}$$

where $\widehat{g}_{t,v} = \nabla_{p_v} \overline{J_t}(p \mid \widehat{w}_t)\big|_{p=p_t} + \xi_t$. This update has the exact mirror form stated in Proposition 1.

Step 3: Final Weight Update ($\mathcal{D}_{train}$) Using the updated routing p_{t+1} and a training minibatch of n examples (x_i, y_i) from $\mathcal{D}_{\text{train}}$, we take a loss-descent step from w_t:

$$w_{t+1} = w_t - \frac{\alpha_t}{n}\sum_{i=1}^{n} \nabla_w L_i^{\text{train}}(w_t, p_{t+1}). \tag{10}$$

Here, α_t and η_t are the weight and architecture step sizes. Algorithm 1 and Figure 3 summarize the sequential updates. We implement the SNN in BrainCog (Zeng et al., 2023) and train weights using STBP with surrogate spike gradients. Each update uses the preceding step's values; routing trajectories illustrate the resulting search dynamics.

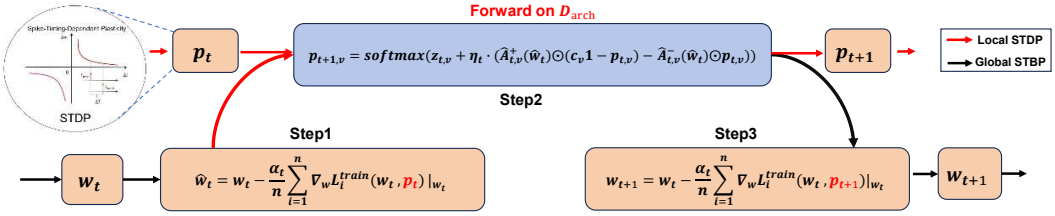


Figure 3: **Three-stage task-conditioned update.** Step 1 computes supervised lookahead weights $\widehat{w}_t$ under p_t. Step 2 freezes them, constructs label-free J_t on $\mathcal{D}_{\text{arch}}$, and takes an STDP mirror step. Step 3 independently resamples training data and updates the original w_t under p_{t+1}.

Analysis of the decoupled DNASNet scheme. Under local smoothness and boundedness assumptions with respect to w, a first-order Taylor expansion of the conditional expected potential $\overline{J}_t$ around $w = w_t$ gives

$$\overline{J}_t(p_t \mid \widehat{w}_t) = \overline{J}_t(p_t \mid w_t) + \alpha_t \left\langle \nabla_w[-\overline{J}_t(p_t \mid w_t)], \nabla_w \frac{1}{n} \sum_{i=1}^{n} L_i^{\text{train}}(w_t, p_t) \right\rangle + \mathcal{O}(\alpha_t^2). \quad (11)$$

When the training-loss and conditional-energy gradients align, the smoothed potential increases to first order, changing the evidence driving subsequent routing updates toward local balance. Observed spike statistics need not be differentiable; assumptions and derivations appear in Appendix A.3.

Algorithm 1 DNASNet search algorithm

Input: $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{arch}}$, minibatch size n, maximum iterations T
Initialize weights w_0 and architecture logits z_0; set $p_0 = \text{GroupedSoftmax}(z_0)$
for $t = 0, 1, \ldots, T-1$ **do**
    $\{x_{\text{train}}, y_{\text{train}}\} \leftarrow \text{SampleMiniBatch}(\mathcal{D}_{\text{train}}, n)$
    $\{x_{\text{arch}}, y_{\text{arch}}\} \leftarrow \text{SampleMiniBatch}(\mathcal{D}_{\text{arch}}, n)$
    Compute intermediate weights $\hat{w}_{(t)}(p)$ by Eq. 8
    Update architecture routing probabilities $p_{(t+1),v}$ via STDP by Eq. 9
    Update network weights $w_{(t+1)}$ by Eq. 10
end for
Output: Network weights w_T and routing probabilities p_T

Under mild conditions, DNASNet can be shown to have stable architecture probabilities and to attain first-order stationarity for the training objective. The theorem statements and proofs are given in Appendix B.

4 Experimental Results for DNASNet

We evaluate DNASNet on three static benchmarks, CIFAR-10 and CIFAR-100 (Krizhevsky, 2009) and ImageNet-1K (Russakovsky et al., 2015), and two event-based benchmarks, DVS-CIFAR-10 (Li et al., 2017) and DVS-Gesture (Amir et al., 2017). We also evaluate applicability beyond vision on the SHD speech dataset (Cramer et al., 2022), reporting results only in the ablation study because we did not extensively tune its hyperparameters. Appendix C details its search-cell design. All image datasets use the search space in Section 3.2. Existing differentiable SNN search methods use specialized search spaces (Che et al., 2022; Man et al., 2024). We implement SNN-DARTS with the same candidate cells and operations as DNASNet, using first- or second-order DARTS updates (Liu et al., 2019). DNASNet's routing probability p lies on the node simplex, whereas SNN-DARTS's architecture parameter α is a slack variable on each edge. DNASNet outperforms the listed SNN-NAS methods on CIFAR-10 without validation-loss routing updates and achieves low spike counts without explicitly optimizing them. We assess discrete-architecture performance, timing-rule ablations, search cost, and inference activity. Appendix C provides experimental details.

For reproducibility, we release DNASNet and SNN-DARTS code, hyperparameters, and random seeds.

4.1 Convergence and Sparsity of DNASNet

We first examine the effects of DNASNet on convergence and sparsity. Figure 4 shows supernetwork validation-accuracy curves during DNASNet and SNN-DARTS search on CIFAR-10 across multiple random seeds. Here, *soft* retains the continuous mixture of candidates in the supernetwork, while *hard* is the supernetwork discretized by pruning rules. Discretization reduces shared-weight validation accuracy, without retraining, from 62.14% to 11.77% for SNN-DARTS and from 74.33% to 41.24% for DNASNet. These results indicate that DNASNet optimizes the supernetwork more effectively under the weight-sharing objective and incurs substantially less performance loss upon discretization.

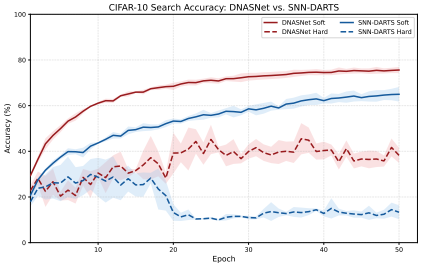


Figure 4: Soft/hard search accuracy.

4.2 Retraining Performance

For DNASNet and SNN-DARTS, we search with four random-seed sets, select the architecture with the highest validation accuracy, retrain from scratch with three seeds, and report the mean and standard deviation of final-epoch accuracy. Some experiments report single-run results. Table 1 compares DNASNet and SNN-DARTS with published SNNs: Spike-driven Transformer (Yao et al., 2023), STBP-tdBN (Zheng et al., 2021), SEW-ResNet (Fang et al., 2021a), and Spikformer (Zhou et al., 2023), and SNN-NAS methods: SNASNet (Kim et al., 2022), Auto-Spikformer (Che et al., 2024a), ES-NAS (Yan et al., 2024), SpikeDHS (Che et al., 2022), AutoSNN (Na et al., 2022), MA-DARTS (Man et al., 2024) and NeuEvo (Shen et al., 2023). Random and first-order SNN-DARTS (Liu et al., 2019) share DNASNet's search space. Published results follow the cited studies, except local reproductions (†), the CIFAR-10 STBP-tdBN row from AutoSNN (Na et al., 2022), and ImageNet SEW-ResNet from NeuEvo (Shen et al., 2023). We run search and evaluation independently per dataset except ImageNet, where, as in DARTS, we retrain the best CIFAR-10 architecture to evaluate transferability and save computation. DNASNet outperforms the listed SNN search baselines in static datasets while maintaining low inference spike counts. Appendix C reports full-pipeline CIFAR-10 repetitions with new seeds.

4.3 Ablating Timing Evidence

To isolate timing, co-activity, and firing-rate effects, we replace the Step 2 routing rule in Section 3.4. *Hebb* removes the exponential LTP/LTD kernels from $\widehat{A}^{\pm}_{t,v}(w)$ and uses only pre- and post-operation firing-rate products. *Rate-only* rewards only high firing rates at candidate-operation outputs. *Random* searches routing probabilities using normalized random perturbations. Table 2 reports retraining results for all four search rules. In our experiments, Hebb and rate-only tend to collapse to architectures containing only average pooling and skip connections. Given their expected poor retraining performance, we did not repeat retraining with multiple seeds. STDP substantially outperforms random perturbations across dataset types, demonstrating timing information's essential role in search. Appendix A.1 details all four rules at operation level. Figure 5(a)–(b) shows the corresponding retraining curves on CIFAR-10 and DVS-Gesture. With matched settings on CIFAR-10/100, DNASNet reduces memory by approximately 20–22% and runtime by 25–51% relative to SNN-DARTS. On DVS-Gesture, all methods share the same peak memory because the temporal-backpropagation graph becomes the computational bottleneck at long time steps. Figure 5(c)–(d) shows layer-wise firing rates as an additional indicator of activity-related cost. Appendix E reports results for further batch sizes and depths, and Appendix D presents an STDP sensitivity analysis.

Table 1: Comparison of DNASNet with published SNNs. Horizontal rules separate handcrafted architectures, SNN-NAS methods, and our baselines and method (top to bottom) within each dataset. **Spikes**: Mean spike count per example during inference; –: Not listed. †: Locally reproduced using the same training settings. ‡: Best result among five architectures randomly sampled from the search space and trained from scratch using the same random seed.

Dataset	Method	Architecture	T	Parameters	Spikes	Accuracy (%)
CIFAR-10	Spike-driven Transformer	Spike-driven Transformer	4	–	–	95.60
	STBP-tdBN	ResNet-19	8	15M	1246K	93.07
	SEW-ResNet†	SEW-ResNet-34	4	25M	260K	94.72
	SNASNet	SNASNet-Bw	8	–	–	94.12
	MA-DARTS	MA-DARTS	2	3.011M	–	94.40
	Auto-Spikformer	Auto-Spikformer $\mathcal{S}_{T_{l2}}$	4	9M	–	96.38
	ES-NAS	Searched	3	23M	–	94.64
	SpikeDHS	CLA(n3c5) + DGS	6	14M	–	95.50
	AutoSNN	AutoSNN(C=128)	8	21M	310K	93.15
	NeuEvo	NeuEvo	4	25M	500K	96.43
	Random baseline‡	Random	4	25M	1450K	94.21
	SNN-DARTS	SNN-DARTS(first)	4	25M	675K	95.37 ± 0.11
	DNASNet	**DNASNet**	**4**	**11M**	**583K**	**95.48**
	DNASNet	**DNASNet**	**4**	**25M**	**610K**	**96.51±0.10**
CIFAR-100	Spike-driven Transformer	Spike-driven Transformer	4	–	–	78.40
	SEW-ResNet†	SEW-ResNet-34	4	25M	339K	77.03
	SNASNet	SNASNet-Bw	5	–	–	73.04
	ES-NAS	Searched	3	27M	–	74.78
	SpikeDHS	CLA(n3s1) + DGS	6	14M	–	76.25
	MA-DARTS	MA-DARTS	2	3.231M	–	76.52
	Auto-Spikformer	Auto-Spikformer $\mathcal{S}_{T_{l2}}$	4	9M	–	77.05
	AutoSNN	AutoSNN(C=64)	8	5M	326K	69.16
	NeuEvo	NeuEvo	4	25M	600K	77.72
	Random baseline‡	Random	4	25M	1480K	75.00
	SNN-DARTS	SNN-DARTS(first)	4	25M	440K	75.28 ± 0.21
	DNASNet	**DNASNet**	**4**	**11M**	**530K**	**78.21**
	DNASNet	**DNASNet**	**4**	**25M**	**589K**	**78.57±0.11**
DVS-CIFAR-10	Spike-driven Transformer	Spike-driven Transformer	16	–	–	80.00
	STBP-tdBN	ResNet-19	10	–	–	67.80
	AutoSNN	AutoSNN(C=16) + stem	20	–	1269K	72.50
	NeuEvo	NeuEvo	10	–	–	84.17
	Random baseline‡	Random	10	19M	–	79.10
	DNASNet	**DNASNet**	**10**	**19M**	–	**81.53±1.18**
ImageNet-1K	Spikformer	Spiking Transformer-6-512	4	23M	–	72.46
	Spike-driven Transformer	Spiking Transformer-6-512	4	23M	–	74.11
	SEW-ResNet	SEW-ResNet-34	4	–	–	67.23
	NeuEvo	NeuEvo	4	26M	–	68.74
	SpikeDHS	CLA-large + DGS	6	58M	–	68.64
	DNASNet	**DNASNet**	**4**	**22M**	–	**73.27**

4.4 Optional Bilinear Integration during Retraining

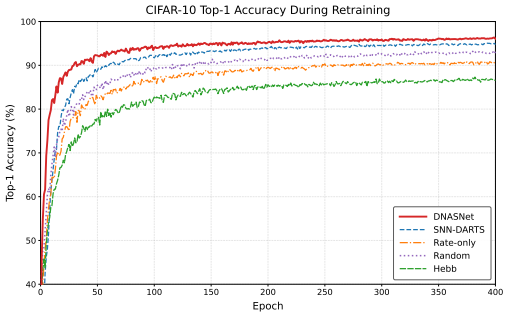
Inspired by bilinear dendritic integration (Li et al., 2014) and our multi-input node topology, we introduce a broadly applicable, low-cost retraining enhancement. It combines selected-path outputs Y_1^t, Y_2^t elementwise:

$$O^t = Y_1^t + Y_2^t + k \cdot (Y_1^t \odot Y_2^t). \tag{12}$$

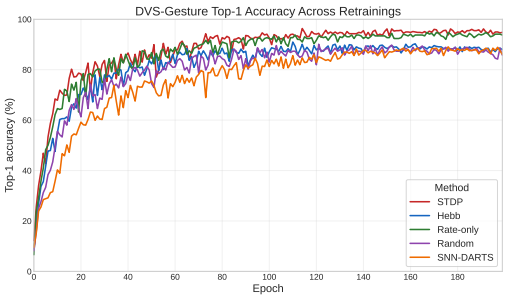
STBP learns k at each of the three integration nodes in every cell. Appendix E.2 gives the results, gradients, and ablations. Accuracy improves by 0.22, 0.28, and 0.13 percentage points on CIFAR-10, CIFAR-100, and ImageNet-1K. All preceding performance results exclude this optional enhancement.

Table 2: Search-rule ablations. Cost: search hours / peak GB on one NVIDIA A800. Accuracy: retraining accuracy.

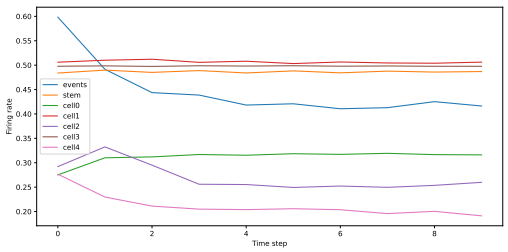
Dataset	Architecture (search rule)	T	Parameters (M)	Spikes (K)	Cost(h/GB)	Accuracy (%)
CIFAR-10	SNN-DARTS	4	24.6	675	5.53/22.40	95.37 ± 0.11
	DNASNet (Random)	4	24.0	1415	3.50/17.16	93.43
	DNASNet (Rate-only)	4	24.0	1586	2.63/17.16	91.17
	DNASNet (Hebb)	4	23.9	1580	2.68/17.17	87.90
	DNASNet (STDP)	4	24.6	610	2.70/18.00	96.51 ± 0.10
CIFAR-100	SNN-DARTS	4	24.6	440	3.30/14.92	75.28 ± 0.21
	DNASNet (Random)	4	24.0	1445	2.46/11.33	66.47
	DNASNet (Rate-only)	4	24.0	1549	2.40/11.33	55.25
	DNASNet (Hebb)	4	23.9	1568	2.43/11.34	56.27
	DNASNet (STDP)	4	24.6	589	2.46/11.67	78.57 ± 0.11
DVS-Gesture	SNN-DARTS	10	0.15	759	1.14/59.69	89.76
	DNASNet (Random)	10	0.15	651	0.48/59.69	93.92
	DNASNet (Rate-only)	10	0.13	859	0.50/59.69	94.97
	DNASNet (Hebb)	10	0.13	856	0.48/59.69	91.49
	DNASNet (STDP)	10	0.15	483	0.61/59.69	97.05 ± 0.25
SHD	DNASNet (Random)	50	0.93	2.14	2.07/1.87	87.18
	DNASNet (STDP)	50	0.93	1.85	2.12/1.87	88.48 ± 0.50



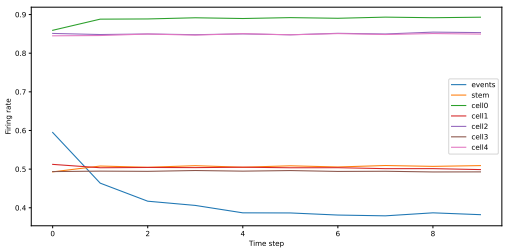
(a) CIFAR-10 retraining



(b) DVS-Gesture retraining



(c) DNASNet: mean firing rates



(d) SNN-DARTS: mean firing rates

Figure 5: **Retraining and inference activity.** (a)–(b) CIFAR-10 and DVS-Gesture test accuracy after retraining architectures found by the five search rules. (c)–(d) DNASNet and SNN-DARTS layer-wise firing rates per inference window. Deeper DNASNet cells have lower activation density, consistent with lower per-example spike activity.

5 Conclusion

DNASNet's biologically plausible decoupled search discovers high-performing SNNs by updating routing probabilities solely from task-conditioned spike timing at fixed network states, without an explicit architecture loss. Surprisingly, its spike counts rival known efficient networks despite using no efficiency regularizer during search.

Reproducibility Statement

Sections 3–4 describe the algorithm and evaluation. Appendices A–B state theoretical assumptions and proofs; Appendix C details architectures, hyperparameters, and training protocols.

Statement on AI Use

Generative AI assisted with wording, spelling, and grammar, and with checking mathematical proofs. The core ideas, methods, and contributions were conceived independently; AI did not generate experimental data or determine the reported results.

References

Yesmine Abdennadher, Giovanni Perin, Riccardo Mazzieri, Jacopo Pegoraro, and Michele Rossi. LightSNN: Lightweight architecture search for sparse and accurate spiking neural networks. In *2025 International Conference on Advanced Machine Learning and Data Science (AMLDS)*. IEEE, 2025. doi: 10.1109/AMLDS63918.2025.11159365. URL https://arxiv.org/abs/2503.21846.

Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo Di Nolfo, Tapan Nayak, Alexander Andreopoulos, Guillaume Garreau, Marcela Mendoza, et al. A low power, fully event-based gesture recognition system. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7243–7252, 2017.

Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

Guo-qiang Bi and Mu-ming Poo. Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *Journal of Neuroscience*, 18 (24):10464–10472, 1998.

Natalia Caporale and Yang Dan. Spike timing–dependent plasticity: a hebbian learning rule. *Annu. Rev. Neurosci.*, 31(1):25–46, 2008. doi: 10.1146/annurev.neuro.31.060407.125639.

Kaiwei Che, Luziwei Leng, Kaixuan Zhang, Jianguo Zhang, Qinghu Meng, Jie Cheng, Qinghai Guo, and Jianxing Liao. Differentiable hierarchical and surrogate gradient search for spiking neural networks. *Advances in Neural Information Processing Systems*, 35:24975–24990, 2022.

Kaiwei Che, Zhaokun Zhou, Jun Niu, Zhengyu Ma, Wei Fang, Yanqi Chen, Shuaijie Shen, Li Yuan, and Yonghong Tian. Auto-Spikformer: Spikformer architecture search. *Frontiers in Neuroscience*, 18:1372257, 2024a.

Kaiwei Che, Zhaokun Zhou, Li Yuan, Jianguo Zhang, Yonghong Tian, and Luziwei Leng. Spatial-temporal search for spiking neural networks. *arXiv preprint arXiv:2410.18580*, 2024b. doi: 10.48550/arXiv.2410.18580. URL https://arxiv.org/abs/2410.18580.

Benjamin Cramer, Yannik Stradmann, Johannes Schemmel, and Friedemann Zenke. The heidelberg spiking data sets for the systematic evaluation of spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 33(7):2744–2757, 2022. doi: 10.1109/TNNLS.2020.3044364.

Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, S. H. Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018. doi: 10.1109/MM.2018.112130359.

Niklas Dexheimer, Sascha Gaudlitz, and Johannes Schmidt-Hieber. Spike-timing-dependent Hebbian learning as noisy gradient descent. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-3555. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/994a9663c3bef2439e605acf340f3bef-Abstract-Conference.html. Revised preprint: arXiv:2505.10272v3, 13 January 2026.

Wei Fang, Zhaofei Yu, Yanqi Chen, Tiejun Huang, Timothée Masquelier, and Yonghong Tian. Deep residual learning in spiking neural networks. *Advances in neural information processing systems*, 34:21056–21069, 2021a.

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

Robert Gütig, Ranit Aharonov, Stefan Rotter, and Haim Sompolinsky. Learning input correlations through nonlinear temporally asymmetric Hebbian plasticity. *Journal of Neuroscience*, 23(9):3697–3714, 2003. doi: 10.1523/JNEUROSCI.23-09-03697.2003. URL https://www.jneurosci.org/content/23/9/3697.

Ilyass Hammouamri, Ismail Khalfaoui-Hassani, and Timothée Masquelier. Learning delays in spiking neural networks using dilated convolutions with learnable spacings. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/4df1cc5a7528b7197ad8ae76ff30107a-Paper-Conference.pdf.

Tingting Jiang, Qi Xu, Xuming Ran, Jiangrong Shen, Pan Lv, Qiang Zhang, and Gang Pan. Adaptive deep spiking neural network with global-local learning via balanced excitatory and inhibitory mechanism. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/4ec43957eda1126ad4887995d05fae3b-Abstract-Conference.html.

Youngeun Kim, Yuhang Li, Hyoungseob Park, Yeshwanth Venkatesha, and Priyadarshini Panda. Neural architecture search for spiking neural networks. In *European conference on computer vision*, pp. 36–56. Springer, 2022.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980.

Jyrki Kivinen and Manfred K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, 1997. doi: 10.1006/inco.1996.2612.

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

Andreea Lazar, Gordon Pipa, and Jochen Triesch. SORN: A self-organizing recurrent neural network. *Frontiers in Computational Neuroscience*, 3:23, 2009. doi: 10.3389/neuro.10.023.2009. URL https://www.frontiersin.org/journals/computational-neuroscience/articles/10.3389/neuro.10.023.2009/full.

Hayeon Lee, Eunyoung Hyung, and Sung Ju Hwang. Rapid neural architecture search by learning to generate graphs from datasets. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2107.00860.

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.

Liam Li, Mikhail Khodak, Maria-Florina Balcan, and Ameet Talwalkar. Geometry-aware gradient algorithms for neural architecture search. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2004.07802.

Songting Li, Nan Liu, Xiao-hui Zhang, Douglas Zhou, and David Cai. Bilinearity in spatiotemporal integration of synaptic inputs. *PLoS computational biology*, 10(12):e1004014, 2014. doi: 10.1371/journal.pcbi.1004014.

Fangxin Liu, Wenbo Zhao, Yongbiao Chen, Zongwu Wang, Tao Yang, and Li Jiang. SSTDP: Supervised spike timing dependent plasticity for efficient spiking neural network training. *Frontiers in Neuroscience*, 15:756876, 2021. doi: 10.3389/fnins.2021.756876.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1eYHoC5FX.

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

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.

Yilei Man, Linhai Xie, Shushan Qiao, Yumei Zhou, and Delong Shang. Differentiable architecture search with multi-dimensional attention for spiking neural networks. *Neurocomputing*, 601:128181, 2024. doi: 10.1016/j.neucom.2024.128181.

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

Milad Mozafari, Mohammad Ganjtabesh, Abbas Nowzari-Dalini, Simon J. Thorpe, and Timothée Masquelier. Bio-inspired digit recognition using reward-modulated spike-timing-dependent plasticity in deep convolutional networks. *Pattern Recognition*, 94:87–95, 2019. doi: 10.1016/j.patcog.2019.05.015.

Byunggook Na, Jisoo Mok, Seongsik Park, Dongjin Lee, Hyeokjun Choe, and Sungroh Yoon. AutoSNN: Towards energy-efficient spiking neural networks. In *International conference on machine learning*, pp. 16253–16269. PMLR, 2022.

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.

Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784):607–617, 2019. doi: 10.1038/s41586-019-1677-2.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y. URL https://www.image-net.org/challenges/LSVRC/2015/.

Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy. Going deeper in spiking neural networks: Vgg and residual architectures. *Frontiers in Neuroscience*, 13:95, 2019. doi: 10.3389/fnins.2019.00095.

Guobin Shen, Dongcheng Zhao, Yiting Dong, and Yi Zeng. Brain-inspired neural circuit evolution for spiking neural networks. *Proceedings of the National Academy of Sciences*, 120(39):e2218173120, 2023. doi: 10.1073/pnas.2218173120.

Xinyu Shi, Zecheng Hao, and Zhaofei Yu. SpikingResformer: Bridging ResNet and vision transformer in spiking neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5610–5619, 2024.

M. C. W. van Rossum, G. Q. Bi, and G. G. Turrigiano. Stable Hebbian learning from spike timing-dependent plasticity. *Journal of Neuroscience*, 20(23):8812–8821, 2000. doi: 10.1523/JNEUROSCI.20-23-08812.2000. URL https://www.jneurosci.org/content/20/23/8812.

Ziqing Wang, Qidong Zhao, Jinku Cui, Xu Liu, and Dongkuan Xu. AutoST: Training-free neural architecture search for spiking transformers. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2024. URL https://arxiv.org/abs/2307.00293.

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.

Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, Yuan Xie, and Luping Shi. Direct training for spiking neural networks: Faster, larger, better. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 1311–1318, 2019.

Yujie Wu, Rong Zhao, Jun Zhu, Feng Chen, Mingkun Xu, Guoqi Li, Sen Song, Lei Deng, Guanrui Wang, Hao Zheng, Songchen Ma, Jing Pei, Youhui Zhang, Mingguo Zhao, and Luping Shi. Brain-inspired global-local learning incorporated with neuromorphic computing. *Nature Communications*, 13:65, 2022. doi: 10.1038/s41467-021-27653-2.

Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. PC-DARTS: Partial channel connections for memory-efficient architecture search. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJlS634tPr.

Jiaqi Yan, Qianhui Liu, Malu Zhang, Lang Feng, De Ma, Haizhou Li, and Gang Pan. Efficient spiking neural network design via neural architecture search. *Neural Networks*, 173:106172, 2024. doi: 10.1016/j.neunet.2024.106172.

An Yang and Ying Liu. Auto-FSDformer: Searching for fully spike-driven transformer. *Knowledge-Based Systems*, 351:116805, 2026. doi: 10.1016/j.knosys.2026.116805. URL https://www.sciencedirect.com/science/article/pii/S0950705126015315.

Man Yao, Jiakui Hu, Zhaokun Zhou, Li Yuan, Yonghong Tian, Bo Xu, and Guoqi Li. Spike-driven transformer. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-2798. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/ca0f5358dbadda74b3049711887e9ead-Abstract-Conference.html.

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.

Yi Zeng, Dongcheng Zhao, Feifei Zhao, Guobin Shen, Yiting Dong, Enmeng Lu, Qian Zhang, Yinqian Sun, Qian Liang, Yuxuan Zhao, Zhuoya Zhao, Hongjian Fang, Yuwei Wang, Yang Li, Xin Liu, Chengcheng Du, Qingqun Kong, Zizhe Ruan, and Weida Bi. BrainCog: A spiking neural network based, brain-inspired cognitive intelligence engine for brain-inspired AI and brain simulation. *Patterns*, pp. 100789, July 2023.

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.

Zhaokun Zhou, Yuesheng Zhu, Chao He, Yaowei Wang, Shuicheng Yan, Yonghong Tian, and Li Yuan. Spikformer: When spiking neural network meets transformer. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2209.15425.

A Exact Node-Wise Entropy Mirror Ascent and an Explicit STDP Potential

A.1 Event Counts

At architecture iteration t, the network state w and the current routing remain fixed during the architecture forward pass, which is performed without gradient tracking. In Step 2, $w = \widehat{w}^{(t)}$; below,

we omit the iteration superscripts on the firing rates. Let B denote the architecture minibatch size, $\mathcal{C}$ the set of cells that share routing parameters, and $r, s \in \{0, \dots, T-1\}$ the simulation time indices.

For cell c, sample b, source node u, and simulation time index r, let $x_{c,b,u,r} \in [0,1]$ denote the presynaptic firing rate detached from the computation graph. For candidate $i = (e, o)$ with source node $u(i)$ and simulation time index s, let $y_{c,b,i,s} \in [0,1]$ denote the detached candidate output firing rate. The pairwise co-activity strength is

$$R_{c,b,i,r,s} = x_{c,b,u(i),r} y_{c,b,i,s}.$$

Because spike counts are normalized in the implementation, firing rates can in principle serve as a reasonable substitute for counts of spike-event pairs. Given positive amplitudes $A_\pm$ and decay time constants $\tau_\pm$,and the time difference between events $\Delta_{rs} = s - r$, define the asymmetric temporal kernels as

$$K^+_{rs} = A_+ e^{-\Delta_{rs}/\tau_+} \mathbf{1}_{\{\Delta_{rs}>0\}},$$
$$K^-_{rs} = A_- e^{\Delta_{rs}/\tau_-} \mathbf{1}_{\{\Delta_{rs}<0\}}.$$

For each cell, the local timing-evidence vectors $\widehat{\boldsymbol{A}}^{\pm}_{c,t,v}(w) \in \mathbb{R}^{d_v}_+$ are defined coordinate-wise as

$$[\widehat{\boldsymbol{A}}^{+}_{c,t,v}(w)]_i = \frac{1}{B} \sum_{b=1}^{B} \sum_{r,s} R_{c,b,i,r,s} K^+_{rs},$$

$$[\widehat{\boldsymbol{A}}^{-}_{c,t,v}(w)]_i = \frac{1}{B} \sum_{b=1}^{B} \sum_{r,s} R_{c,b,i,r,s} K^-_{rs}.$$

For a shared architecture, we average the evidence over the search cells:

$$\widehat{\boldsymbol{A}}^{\pm}_{t,v}(w) = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \widehat{\boldsymbol{A}}^{\pm}_{c,t,v}(w). \tag{13}$$

These vectors are computed before the logit update and treated as fixed stop-gradient coefficients during that update.

Hebb: The Hebb rule ignores timing-dependent probability decay and considers only pairwise co-activity strength. Its update is written coordinate-wise as

$$[\widehat{\boldsymbol{A}}_{t,v}(w)]_i = \frac{1}{|\mathcal{C}|} \frac{1}{B} \sum_{c \in \mathcal{C}} \sum_{b=1}^{B} \sum_{r,s} R_{c,b,i,r,s}. \tag{14}$$

rate-only: The rate-only rule is simpler: it considers only the firing rate after the operation. Its update strength is written coordinate-wise as

$$[\widehat{\boldsymbol{A}}_{t,v}(w)]_i = \frac{1}{|\mathcal{C}|} \frac{1}{B} \sum_{c \in \mathcal{C}} \sum_{b=1}^{B} \sum_{s} y_{c,b,i,s}. \tag{15}$$

Random: To test the contribution of timing evidence, the random perturbation baseline directly replaces the timing-evidence update with i.i.d. Gaussian noise:

$$[\widehat{\boldsymbol{A}}^{\pm}_{t,v}(w)]_i = \xi_{t,v,i} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1). \tag{16}$$

A.2 Derivations of Propositions 1 and 2

This section proves the optimization results underlying the algorithm design, using the classical exponentiated-gradient and entropy-mirror formulations (Kivinen & Warmuth, 1997; Beck & Teboulle, 2003).

Proposition 1: Node-wise entropy mirror ascent.

For each routing group v updated according to Eq. 4, finite logits and $\eta_t > 0$ yield the probability update

$$p_{t+1,v} = \underset{q_v \in \Delta^{d_v - 1}}{\arg\max} \left\{ \eta_t \langle q_v, \widehat{g}_{t,v} \rangle - D_{\mathrm{KL}}(q_v \| p_{t,v}) \right\}. \tag{17}$$

Equivalently,

$$p_{t+1,v} = \frac{p_{t,v} \odot \exp(\eta_t \widehat{g}_{t,v})}{\langle p_{t,v}, \exp(\eta_t \widehat{g}_{t,v}) \rangle}. \tag{18}$$

Considering all feedforward and feedback groups jointly gives

$$p_{t+1} = \arg\max_{q \in \mathcal{P}} \left\{ \eta_t \sum_v \langle q_v, \widehat{g}_{t,v} \rangle - D_{\mathcal{P}}(q \| p_t) \right\}. \tag{19}$$

Proof. Eq. 4 and the node-wise softmax give

$$p_{t+1,v} = \frac{p_{t,v} \odot \exp(\eta_t \widehat{g}_{t,v})}{\langle p_{t,v}, \exp(\eta_t \widehat{g}_{t,v}) \rangle},$$

which proves Eq. 18. For the variational form, fix a routing group and maximize the following objective over the simplex:

$$\Phi(q) = \eta_t \langle q, g \rangle - D_{\mathrm{KL}}(q \| p)$$

For $p \in \mathrm{int}(\Delta)$, the negative KL term is strictly concave, so the maximizer is unique. The KKT stationarity conditions give $q_i \propto p_i e^{\eta_t g_i}$, which yields Eq. 18 after normalization. Both the feasible set and the KL divergence decompose by routing group, giving the result on the product space. Once the evidence is frozen, this decomposition holds even when the neural dynamics are coupled. Finally, for any scalar a, $\mathrm{softmax}(z_v + a\mathbf{1}) = \mathrm{softmax}(z_v)$, so a common shift of all scores within a group leaves the routing unchanged. □

Proposition 2: The STDP utility is the gradient of an explicit concave potential.

For the STDP utility in Eq. 3 and the potential in Eq. 6, we have

$$\widehat{g}_t = \nabla_p J_t(p_t \mid w). \tag{20}$$

Proof. Holding the timing evidence fixed and differentiating Eq. 6 node-wise gives

$$\nabla_{p_v} J_t(p \mid w) = c_v \widehat{\boldsymbol{A}}^+_{t,v}(w) - \left(\widehat{\boldsymbol{A}}^+_{t,v}(w) + \widehat{\boldsymbol{A}}^-_{t,v}(w) \right) \odot p_v,$$

which equals the utility in Eq. 3. The Hessian is block diagonal; because the evidence vectors are nonnegative, every block is negative semidefinite:

$$\nabla^2_{p_v p_v} J_t = -\operatorname{Diag}\left(\widehat{\boldsymbol{A}}^+_{t,v}(w) + \widehat{\boldsymbol{A}}^-_{t,v}(w) \right) \preceq 0, \tag{21}$$

Thus, $J_t(\cdot \mid w)$ is concave on $\mathcal{P}$. □

For a random architecture minibatch, the noise decomposes as follows. Let $\mathcal{F}_t$ denote the information available before the architecture minibatch is sampled at iteration t. In the three-stage iteration in Section 3.4, the one-step supervised lookahead state $\widehat{w}_t$ is already fixed at this point. Define the conditional mean timing evidence as

$$\overline{\boldsymbol{A}}^{\pm}_{t,v} = \mathbb{E}\left[\widehat{\boldsymbol{A}}^{\pm}_{t,v}(\widehat{w}_t) \mid \mathcal{F}_t \right],$$

and define the conditional population potential as

$$\overline{J}_t(p \mid \widehat{w}_t) = \sum_v \left[c_v (\overline{\boldsymbol{A}}^+_{t,v})^\top p_v - \frac{1}{2} (\overline{\boldsymbol{A}}^+_{t,v} + \overline{\boldsymbol{A}}^-_{t,v})^\top (p_v \odot p_v) \right]. \tag{22}$$

Define $\xi_t = \widehat{g}_t - \nabla_p \overline{J}_t(p_t \mid \widehat{w}_t)$. Then

$$\widehat{g}_t = \nabla_p \overline{J}_t(p_t \mid \widehat{w}_t) + \xi_t, \qquad \mathbb{E}[\xi_t \mid \mathcal{F}_t] = 0. \tag{23}$$

Proof. Conditioned on $\mathcal{F}_t$, p_t, $\widehat{w}_t$, and c_v are fixed. By the linearity of conditional expectation,

$$\begin{aligned} \mathbb{E}[\widehat{g}_{t,v} \mid \mathcal{F}_t] &= \overline{\boldsymbol{A}}^+_{t,v} \odot (c_v \mathbf{1} - p_{t,v}) - \overline{\boldsymbol{A}}^-_{t,v} \odot p_{t,v} \\ &= \nabla_{p_v} \overline{J}_t(p_t \mid \widehat{w}_t), \end{aligned}$$

which proves the result. □

A.3 A Gradient-Alignment Interpretation of Supervised Lookahead

Fix the current routing p_t and lookahead minibatch $\mathcal{B}_t^{\text{look}}$, and write $d_t = \nabla_w \mathcal{L}_t^{\text{look}}(w_t, p_t)$. Eq. (8) then gives $\widehat{w}_t - w_t = -\alpha_t d_t$. The analysis uses the conditional expected potential $\overline{J}_t(p \mid w)$ from Proposition 2. In the Taylor comparison, we hold the conditioning information $\mathcal{F}_t$ and the conditional distribution of the architecture minibatch fixed, varying only the network-state parameter w in the family of functions. We additionally assume that this function is twice continuously differentiable with respect to w in a neighborhood containing the line segment between w_t and $\widehat{w}_t$, and that its Hessian and d_t are locally bounded. This assumption concerns the smooth family of functions formed from expected timing evidence; it does not follow merely from the weights being continuous-valued, nor does it require the observed discrete spike statistics to be differentiable.

By Taylor's formula,

$$\begin{aligned}\overline{J}_t(p_t \mid \widehat{w}_t) - \overline{J}_t(p_t \mid w_t) &= \nabla_w \overline{J}_t(p_t \mid w_t)^\top (\widehat{w}_t - w_t) + R_t \\ &= -\alpha_t \langle \nabla_w \overline{J}_t(p_t \mid w_t), d_t \rangle + R_t \\ &= \alpha_t \langle \nabla_w [-\overline{J}_t(p_t \mid w_t)], d_t \rangle + R_t,\end{aligned}$$

where $|R_t| \le \frac{1}{2} M_J \alpha_t^2 \|d_t\|_2^2 = \mathcal{O}(\alpha_t^2)$, and M_J is an upper bound on the Hessian operator norm in this neighborhood. This yields Eq. (11). When the inner product is positive and the step size is sufficiently small, supervised lookahead increases this smooth conditional potential. This result characterizes the local change in the state that generates the evidence; it does not guarantee an increase in the routing probability of every candidate. The actual architecture update still holds $\widehat{w}_t$ and the observed evidence fixed and does not compute $\partial \widehat{w}_t / \partial p$.

B Architecture Stability, Training-Gradient Convergence, and Online Regret

This section establishes routing stability for the exact STDP mirror update, first-order stationarity of persistent training under a stochastic-gradient abstraction, and a finite-horizon online regret bound. The first result relies on bounded utilities and step-size conditions, whereas the second requires a smooth surrogate objective, unbiased training gradients, and finite total variation of the architecture. The training result does not directly cover AdamW with adaptive preconditioning, nor does it imply convergence of the parameters to a single point or global optimality for supervised NAS.

B.1 Stability of Architecture Updates

Theorem 3 (Stability of architecture updates)**.** *Assume that the node-wise stochastic STDP utilities are uniformly bounded, with $\|\widehat{g}_{t,v}\|_\infty \le G_v$, and that the deterministic architecture step sizes satisfy $\eta_t \ge 0$ and $\eta_t \to 0$. Then*

$$\|p_{t+1} - p_t\|_1 \le 2\eta_t \sum_{v \in \mathcal{V}} G_v \longrightarrow 0. \tag{24}$$

If, in addition, $\sum_t \eta_t < \infty$, then $\sum_t \|p_{t+1} - p_t\|_1 < \infty$, and p_t converges almost surely to a random limiting routing distribution p_∞. If both the feedforward and feedback parsing rules have strict selection gaps at the limit, the complete discrete cell genotype eventually remains unchanged.

These results describe stability along a single stochastic trajectory; they do not guarantee identical limiting routing distributions across random seeds.

To prove Theorem 3, consider the following bound.

For the exact logit update in Eq. 4 with $\eta_t \ge 0$, suppose that finite constants G_v satisfy $\|\widehat{g}_{t,v}\|_\infty \le G_v$ almost surely for all t, v. Then

$$\|p_{t+1,v} - p_{t,v}\|_1 \le 2\eta_t G_v. \tag{25}$$

Proof. Define the path

$$\phi_v(s) = \operatorname{softmax}(z_{t,v} + s\widehat{g}_{t,v}), \qquad 0 \le s \le \eta_t.$$

Differentiating the softmax gives

$$\phi_v'(s) = \phi_v(s) \odot (\widehat{g}_{t,v} - \langle \phi_v(s), \widehat{g}_{t,v} \rangle \mathbf{1}) .$$

Therefore,

$$\begin{aligned} \|\phi_v'(s)\|_1 &\le \langle \phi_v(s), |\widehat{g}_{t,v}| + |\langle \phi_v(s), \widehat{g}_{t,v} \rangle| \mathbf{1} \rangle \\ &\le G_v + G_v = 2G_v. \end{aligned}$$

By the fundamental theorem of calculus,

$$\|p_{t+1,v} - p_{t,v}\|_1 = \left\| \int_0^{\eta_t} \phi_v'(s)\, ds \right\|_1 \le \int_0^{\eta_t} \|\phi_v'(s)\|_1\, ds \le 2\eta_t G_v.$$

□

If $\eta_t \to 0$, then

$$\|p_{t+1} - p_t\|_1 \le 2\eta_t \sum_{v \in \mathcal{V}} G_v \longrightarrow 0 \qquad \text{almost surely.} \tag{26}$$

If, in addition, $\sum_{t=0}^{\infty} \eta_t < \infty$, then $\sum_{t=0}^{\infty} \|p_{t+1} - p_t\|_1 < \infty$, and $p_t \to p_\infty$ almost surely for some random variable $p_\infty \in \mathcal{P}$. Indeed, (p_t) is a Cauchy sequence, and the product of finite-dimensional closed simplices is both complete and compact. Hence, p_t converges to some $p_\infty \in \mathcal{P}$.

Corollary 4 (Finite-time genotype stabilization under strict limiting gaps)**.** *Assume that the limiting routing distribution p_∞ in Theorem 3 almost surely has positive selection gaps: between the winning operation and every other operation on each feedforward edge; between the second- and third-highest edge scores at each node with unselected feedforward edges; and between the winning edge–operation pair and every other candidate in each feedback group. Then there exists an almost surely finite T_0 such that the complete feedforward and feedback topology remains unchanged for all $t \ge T_0$. If score rankings also determine the storage order of the genotype, the corresponding comparisons must likewise have strict limiting gaps.*

Proof. Fix a sample path along which the routing distribution converges and the limiting gaps are positive. Because there are finitely many comparisons, the minimum limiting gap γ is positive. Candidate scores are probability coordinates, and each feedforward edge score is the maximum candidate probability on that edge. Both are continuous in p, with their changes controlled by $\|p - p_\infty\|_\infty$. By convergence, there exists a finite T_0 after which every relevant score differs from its limit by less than $\gamma/4$. Each score difference therefore deviates from its limiting value by less than $\gamma/2$ and retains its sign. The winning feedforward operations, selected feedforward edges, and feedback target–operation pairs consequently remain unchanged. The same argument applies to comparisons that determine the storage order. Such sample paths have probability one. □

B.2 Convergence of STBP Training Gradients under Persistent Updates

For persistent classifier training, the one-step supervised lookahead state is

$$\widehat{w}_t = w_t - \alpha_t \left[\nabla_w F(w_t, p_t) + \chi_{t+1} \right], \tag{27}$$

where χ_{t+1} is the stochastic-gradient noise from the lookahead training minibatch $\mathcal{B}_t^{\text{look}}$. This temporary state is used to construct STDP statistics on an independently sampled architecture minibatch $\mathcal{B}_t^{\text{arch}}$. The architecture update is

$$z_{t+1,v} = z_{t,v} + \eta_t \widehat{g}_{t,v}, \qquad p_{t+1} = \text{GroupedSoftmax}(z_{t+1}). \tag{28}$$

Here, $\widehat{g}_{t,v}$ is the frozen-statistics utility in Eq. 3, evaluated at $(p_t, \widehat{w}_t)$; its conditional noise decomposition is given in Proposition 2. After p_{t+1} has been determined, Step 3 uses a training minibatch $\mathcal{B}_t^{\text{train}}$ and updates the original weights w_t:

$$w_{t+1} = w_t - \alpha_t \left[g_t + \psi_{t+1} \right], \qquad g_t := \nabla_w F(w_t, p_{t+1}). \tag{29}$$

These two weight updates share the step symbol α_t, sampling does not require disjoint samples. STDP potentials use look-ahead states $\widehat{w}_t$, while training targets use persistent weights w_t.

Let $\mathcal{G}_t$ be the σ-algebra generated by the information available before sampling in Step 3, including the history, (w_t, p_t), the current lookahead and architecture minibatches, and the resulting $(\widehat{w}_t, p_{t+1})$. The fresh training minibatch in Step 3 is independent of $\mathcal{G}_t$. Under the assumptions of unbiased stochastic gradients and bounded variance,

$$\mathbb{E}[\psi_{t+1} \mid \mathcal{G}_t] = 0, \qquad \mathbb{E}[\|\psi_{t+1}\|_2^2 \mid \mathcal{G}_t] \le \sigma_w^2.$$

Thus, g_t is $\mathcal{G}_t$-measurable, and $\mathbb{E}\langle g_t, \psi_{t+1}\rangle = 0$. The proof does not require the next routing distribution p_{t+2} to be independent of ψ_{t+1}.

Theorem 5 (Convergence of STBP training gradients)**.** *Let $F(w, p)$ be a differentiable surrogate objective that is bounded below, and suppose that STBP uses its stochastic gradients. Assume that the objective is twice continuously differentiable in a neighborhood of a bounded region containing the iterates and the line segments used in the proof, with uniform bounds $\|\nabla_w F\|_2 \le G_w$, $\|\nabla_p F\|_\infty \le G_p$, $\|\nabla^2_{ww} F\|_{\mathrm{op}} \le L_{ww}$, and $\|\nabla^2_{wp} F\|_{2\leftarrow 1} \le L_{wp}$. Assume that the stochastic gradients in Step 3 are conditionally unbiased with uniformly bounded variance, that the deterministic weight step sizes satisfy $\alpha_t > 0$, $\sum_t \alpha_t = \infty$, and $\sum_t \alpha_t^2 < \infty$, and that the total variation of the architecture satisfies $\sum_t \mathbb{E}\|p_{t+1} - p_t\|_1 < \infty$. Under bounded utilities, $\sum_t \eta_t < \infty$ is sufficient for the last condition. Then*

$$\lim_{t\to\infty} \mathbb{E}\big[\|\nabla_w F(w_t, p_{t+1})\|_2^2\big] = 0. \tag{30}$$

The following lemma allows us to deduce first-order stationarity along the entire training-gradient sequence from weighted summability.

Lemma 6 (Sequence lemma with summable perturbations)**.** *Let $0 \le q_t \le Q < \infty$ and $\alpha_t > 0$ satisfy $\sum_t \alpha_t = \infty$ and $\sum_t \alpha_t q_t < \infty$. If a finite constant $K \ge 0$ and a nonnegative summable sequence ε_t satisfy $|q_{t+1} - q_t| \le K\alpha_t + \varepsilon_t$, then $q_t \to 0$.*

Proof. Since $\sum_t \alpha_t = \infty$, we have $\sum_t(\alpha_t + \varepsilon_t) = \infty$. Because $q_t \le Q$ and $\sum_t \varepsilon_t < \infty$,

$$\sum_t (\alpha_t + \varepsilon_t) q_t = \sum_t \alpha_t q_t + \sum_t \varepsilon_t q_t \le \sum_t \alpha_t q_t + Q \sum_t \varepsilon_t < \infty.$$

Moreover,

$$|q_{t+1} - q_t| \le K'\alpha_t + K'\varepsilon_t = K'(\alpha_t + \varepsilon_t), \qquad K' := \max\{K, 1\}.$$

Apply an excursion-interval argument with weights $\alpha_t + \varepsilon_t$. Weighted summability and $\sum_t(\alpha_t + \varepsilon_t) = \infty$ imply $\liminf_t q_t = 0$. If $\limsup_t q_t > 0$, there exists $\delta > 0$ such that $q_t \ge 2\delta$ infinitely often, and each such high point is followed by an index satisfying $q_t \le \delta$. Choose disjoint descent intervals $[r, s)$, each starting at $q_r \ge 2\delta$ and ending at the first $s > r$ for which $q_s \le \delta$. Thus, $q_k > \delta$ for every $k = r, \dots, s-1$, and

$$\delta \le q_r - q_s \le \sum_{k=r}^{s-1} |q_{k+1} - q_k| \le K' \sum_{k=r}^{s-1} (\alpha_k + \varepsilon_k),$$

Hence, $\sum_{k=r}^{s-1}(\alpha_k + \varepsilon_k) \ge \delta/K'$, so the contribution of each interval satisfies

$$\sum_{k=r}^{s-1} (\alpha_k + \varepsilon_k) q_k \ge \delta \sum_{k=r}^{s-1} (\alpha_k + \varepsilon_k) \ge \frac{\delta^2}{K'}$$

Because $\sum_t(\alpha_t + \varepsilon_t)q_t$ is finite, this uniform positive lower bound on the contribution rules out infinitely many disjoint intervals. Therefore, $\limsup_t q_t = 0$, and hence $q_t \to 0$. □

We now prove Theorem 5.

The persistent-update recursion in Eq. 29 satisfies

$$\lim_{t\to\infty} \mathbb{E}\left[\|\nabla_w F(w_t, p_{t+1})\|_2^2\right] = 0. \tag{31}$$

Writing $q_t := \mathbb{E}\|g_t\|_2^2$ and $\Delta^p_{t+1} := \mathbb{E}\|p_{t+2} - p_{t+1}\|_1$, we also have

$$|q_{t+1} - q_t| \le C_w \alpha_t + C_p \Delta^p_{t+1} \longrightarrow 0, \tag{32}$$

where we may choose $C_w = 2G_w L_{ww}\sqrt{G_w^2 + \sigma_w^2}$ and $C_p = 2G_w L_{wp}$, independently of t.

Proof. The proof has four steps.

Step 1: Separate architecture drift from descent under persistent weight updates. Define $F_t := F(w_t, p_{t+1})$ and insert $F(w_{t+1}, p_{t+1})$ to separate architecture drift from the persistent weight update:

$$\begin{aligned} F_{t+1} - F_t = &\underbrace{F(w_{t+1}, p_{t+2}) - F(w_{t+1}, p_{t+1})}_{A_t\text{: architecture drift}} \\ &+ \underbrace{F(w_{t+1}, p_{t+1}) - F(w_t, p_{t+1})}_{B_t\text{: persistent STBP step}}. \end{aligned} \tag{33}$$

Equation 33 does not involve $\widehat{w}_t$, which is used only in the STDP potential in Eqs. 27–28.

For A_t, let $\Delta p_{t+1} = p_{t+2} - p_{t+1}$. Applying the fundamental theorem of calculus along the architecture line segment gives

$$A_t = \int_0^1 \langle \nabla_p F(w_{t+1}, p_{t+1} + s\Delta p_{t+1}), \Delta p_{t+1} \rangle \, ds.$$

By ℓ_∞–ℓ_1 duality and the boundedness of $\nabla_p F$,

$$|A_t| \le G_p \|p_{t+2} - p_{t+1}\|_1. \tag{34}$$

For B_t, L_{ww}-smoothness in w gives

$$\begin{aligned} B_t &\le \langle g_t, w_{t+1} - w_t \rangle + \frac{L_{ww}}{2} \|w_{t+1} - w_t\|_2^2 \\ &= -\alpha_t \|g_t\|_2^2 - \alpha_t \langle g_t, \psi_{t+1} \rangle + \frac{L_{ww}\alpha_t^2}{2} \|g_t + \psi_{t+1}\|_2^2. \end{aligned}$$

Taking expectations conditional on $\mathcal{G}_t$ and using the unbiasedness of the stochastic gradient yields

$$\mathbb{E}[B_t \mid \mathcal{G}_t] \le -\alpha_t \left(1 - \frac{L_{ww}\alpha_t}{2}\right) \|g_t\|_2^2 + \frac{L_{ww}\alpha_t^2}{2} \sigma_w^2. \tag{35}$$

Since $\alpha_t \to 0$, there exists T_1 such that $L_{ww}\alpha_t \le 1$ for all $t \ge T_1$. Combining Eqs. 33, 34, and 35, and then taking total expectations, gives

$$\mathbb{E}[F_{t+1}] \le \mathbb{E}[F_t] - \frac{\alpha_t}{2} \mathbb{E}\|g_t\|_2^2 + G_p \mathbb{E}\|p_{t+2} - p_{t+1}\|_1 + \frac{L_{ww}\sigma_w^2}{2} \alpha_t^2. \tag{36}$$

Step 2: Establish weighted summability of the training-gradient norm. Summing Eq. 36 from T_1 to T and using $F_{T+1} \ge F_{\inf}$ yields

$$\frac{1}{2} \sum_{t=T_1}^{T} \alpha_t \mathbb{E}\|g_t\|_2^2 \le \mathbb{E}[F_{T_1}] - F_{\inf} + G_p \sum_{t=T_1}^{T} \mathbb{E}\|p_{t+2} - p_{t+1}\|_1 + \frac{L_{ww}\sigma_w^2}{2} \sum_{t=T_1}^{T} \alpha_t^2.$$

Both series on the right are finite by the assumptions of the theorem. Letting $T \to \infty$, we obtain

$$\sum_{t=0}^{\infty} \alpha_t \mathbb{E}\|g_t\|_2^2 < \infty. \tag{37}$$

This implies $\liminf_t \mathbb{E}\|g_t\|_2^2 = 0$. Convergence of the entire sequence additionally requires control of its one-step variation.

Step 3: Bound the one-step variation of the training gradient. Let

$$g_{t+1} = \nabla_w F(w_{t+1}, p_{t+2}), \qquad g_t = \nabla_w F(w_t, p_{t+1}).$$

Insert the intermediate gradient $\nabla_w F(w_t, p_{t+2})$:

$$\begin{aligned} g_{t+1} - g_t = &\underbrace{\nabla_w F(w_{t+1}, p_{t+2}) - \nabla_w F(w_t, p_{t+2})}_{C_t\text{: weight change}} \\ &+ \underbrace{\nabla_w F(w_t, p_{t+2}) - \nabla_w F(w_t, p_{t+1})}_{D_t\text{: architecture change}}. \end{aligned}$$

Let $\Delta w_t = w_{t+1} - w_t$. The chain rule gives

$$C_t = \int_0^1 \nabla^2_{ww} F(w_t + s\Delta w_t, p_{t+2})\Delta w_t \, ds,$$

Likewise, with $\Delta p_{t+1} = p_{t+2} - p_{t+1}$, we have

$$D_t = \int_0^1 \nabla^2_{wp} F(w_t, p_{t+1} + s\Delta p_{t+1})\Delta p_{t+1} \, ds.$$

Therefore,

$$\|g_{t+1} - g_t\|_2 \le L_{ww}\|w_{t+1} - w_t\|_2 + L_{wp}\|p_{t+2} - p_{t+1}\|_1.$$

Now use

$$\big|\|a\|_2^2 - \|b\|_2^2\big| = \big|(\|a\|_2 + \|b\|_2)(\|a\|_2 - \|b\|_2)\big| \le (\|a\|_2 + \|b\|_2)\|a - b\|_2.$$

Since $\|g_t\|_2 \le G_w$,

$$\begin{aligned} |q_{t+1} - q_t| &\le 2G_w \mathbb{E}\|g_{t+1} - g_t\|_2 \\ &\le 2G_w L_{ww}\mathbb{E}\|w_{t+1} - w_t\|_2 + 2G_w L_{wp}\Delta^p_{t+1}. \end{aligned} \tag{38}$$

By Eq. 29, Jensen's inequality, and the Cauchy–Schwarz inequality,

$$\begin{aligned} \mathbb{E}\|w_{t+1} - w_t\|_2 &= \alpha_t \mathbb{E}\|g_t + \psi_{t+1}\|_2 \\ &\le \alpha_t \sqrt{\mathbb{E}\|g_t + \psi_{t+1}\|_2^2} \\ &= \alpha_t \sqrt{\mathbb{E}\|g_t\|_2^2 + \mathbb{E}\|\psi_{t+1}\|_2^2} \\ &\le \alpha_t \sqrt{G_w^2 + \sigma_w^2}, \end{aligned}$$

The third equality uses the fact that ψ_{t+1} has zero conditional mean. Substituting into Eq. 38 gives

$$\big|\mathbb{E}\|\nabla_w F(w_{t+1}, p_{t+2})\|_2^2 - \mathbb{E}\|\nabla_w F(w_t, p_{t+1})\|_2^2\big| \le 2G_w L_{ww}\sqrt{G_w^2 + \sigma_w^2}\,\alpha_t + 2G_w L_{wp}\Delta^p_{t+1}.$$

This bound tends to zero because $\alpha_t \to 0$ and finite expected total variation of the architecture implies $\Delta^p_{t+1} \to 0$.

Step 4: Establish first-order stationarity of training. Apply Lemma 6 with $q_t = \mathbb{E}\|g_t\|_2^2$ and $\varepsilon_t = C_p\Delta^p_{t+1}$. Equation 37 gives $\sum_t \alpha_t q_t < \infty$, the step sizes satisfy $\sum_t \alpha_t = \infty$, Eq. 32 provides the required variation bound, and

$$\sum_t \varepsilon_t = C_p \sum_t \Delta^p_{t+1} < \infty.$$

Therefore, $q_t \to 0$, establishing Eq. 31. □

B.3 Finite-Horizon Online Regret for Time-Varying STDP Potentials

The asymptotic theorem uses finite architecture variation to treat routing drift as a summable perturbation to STBP. The finite-horizon guarantee uses the standard entropy-mirror analysis (Kivinen & Warmuth, 1997; Beck & Teboulle, 2003) and does not require this condition; throughout this subsection, we take $\eta_t \equiv \eta > 0$. Let J_t denote the conditional potential obtained at architecture step t, suppressing its frozen state and evidence, and let $g_t = \nabla J_t(p_t)$. Here, g_t is the architecture utility, rather than the training gradient used in the preceding subsection.

Lemma 7 (One-step entropy mirror inequality)**.** *For the exact mirror update*

$$p_{t+1} = \arg\max_{r \in \mathcal{P}} \{\eta\langle r, g_t\rangle - D_{\mathcal{P}}(r\|p_t)\},$$

every comparator $q \in \mathcal{P}$ satisfies

$$\eta\langle g_t, q - p_t\rangle \le D_{\mathcal{P}}(q\|p_t) - D_{\mathcal{P}}(q\|p_{t+1}) + \frac{\eta^2}{2}\sum_v \|g_{t,v}\|_\infty^2. \tag{39}$$

Proof. First-order optimality of the KL-regularized mirror subproblem and the Bregman three-point identity give

$$\eta\langle g_t, q - p_{t+1}\rangle \le D_{\mathcal{P}}(q\|p_t) - D_{\mathcal{P}}(q\|p_{t+1}) - D_{\mathcal{P}}(p_{t+1}\|p_t).$$

Add $\eta\langle g_t, p_{t+1} - p_t\rangle$ to both sides and apply Hölder's inequality node-wise:

$$\eta\langle g_t, p_{t+1} - p_t\rangle \le \eta \sum_v \|g_{t,v}\|_\infty \|p_{t+1,v} - p_{t,v}\|_1.$$

By Pinsker's inequality,

$$D_{\mathcal{P}}(p_{t+1}\|p_t) \ge \frac{1}{2}\sum_v \|p_{t+1,v} - p_{t,v}\|_1^2.$$

For each group, set $a = \|p_{t+1,v} - p_{t,v}\|_1$ and $b = \eta\|g_{t,v}\|_\infty$. Applying $ab - a^2/2 \le b^2/2$ yields Eq. 39. □

Theorem 8 (Static regret against the best fixed architecture distribution)**.** *Let $J_t : \mathcal{P} \to \mathbb{R}$ be any sequence of differentiable concave functions with $g_t = \nabla J_t(p_t)$ and $\|g_{t,v}\|_\infty \le G_v$. For a constant mirror step size $\eta_t \equiv \eta > 0$, every fixed comparator $q \in \mathcal{P}$ satisfies*

$$\sum_{t=1}^{T} [J_t(q) - J_t(p_t)] \le \frac{D_{\mathcal{P}}(q\|p_1)}{\eta} + \frac{\eta}{2}\sum_{t=1}^{T}\sum_v \|g_{t,v}\|_\infty^2. \tag{40}$$

If each $p_{1,v}$ is uniform, then $D_{\mathcal{P}}(q\|p_1) \le D_0 := \sum_v \log d_v$. When $D_0 > 0$ and $S := \sum_v G_v^2 > 0$, choosing $\eta = \sqrt{2D_0/(TS)}$ gives

$$\frac{1}{T}\sum_{t=1}^{T} [J_t(q) - J_t(p_t)] \le \sqrt{\frac{2D_0 S}{T}} = O(T^{-1/2}). \tag{41}$$

Proof. By concavity,

$$J_t(q) - J_t(p_t) \le \langle g_t, q - p_t\rangle.$$

Apply Lemma 7, divide by η, and sum from 1 to T. Telescoping and the nonnegativity of the final KL term yield Eq. 40. For uniform $p_{1,v}$,

$$D_{\mathrm{KL}}(q_v\|p_{1,v}) = \langle q_v, \log q_v\rangle + \log d_v \le \log d_v,$$

Hence, $D_{\mathcal{P}}(q\|p_1) \le D_0$. Bounding the gradient term by TS and minimizing $D_0/\eta + \eta TS/2$ over η gives the stated step size and Eq. 41. □

C Experimental Setup and Reproducibility Details

C.1 Detailed DNASNet Search Space

We consider convolutional spiking neural network cells. Following the evaluation protocols of Liu et al. (2019) and Xu et al. (2020), we evaluate DNASNet in three stages:

- **Stage 1:** During search, we run DNASNet with four random seeds to reduce the variance arising from initialization of the shared-weight network.
- **Stage 2:** We evaluate the best architecture identified in each search run by training it from scratch.
- **Stage 3:** We evaluate the best architecture from Stage 2 more extensively using three different random seeds for initialization.

For completeness, we describe the convolutional spiking neural network search space. Each cell contains 5 nodes: 2 input nodes (0, 1) and 3 intermediate nodes (2, 3, 4). The nodes are ordered, and later nodes can receive the outputs of earlier nodes as inputs. A feedback edge passes a node's output at time t-1 to its target node as an input at time t. Table 3 lists the nine candidate feedforward edges, seven candidate feedback edges, and their competition groups. In the four-operation setting, all edges share $\mathcal{O}$. A feedforward group $v = (\mathrm{f}, j)$ has node j as its target, whereas a feedback group $v = (\mathrm{b}, u)$ has

node u as its source. Thus, $(\mathrm{f}, 3)$ and $(\mathrm{b}, 3)$ correspond to the same physical node but are normalized separately. We search only normal cells and use the fixed reduction-cell structure. The resulting discrete search space therefore contains $\left[\prod_{m=2}^{4} \binom{m}{2} 4^2\right] (3 \cdot 4)(4 \cdot 4) = 14,155,776 \approx 1.42 \times 10^7$ possible architectures. However, because we do not use the relaxation operation in standard DARTS, competition among operations arises naturally at the node level. The number of architectures that may be explored during search is consequently larger, namely $4^{16} \approx 4.3 \times 10^9$.

Table 3: Candidate connections and routing groups in a normal image cell. Each group includes all operations on the listed edges; d_v is calculated for the four-operation setting. Feedforward and feedback groups have different orientations but share the same STDP utility and mirror update.

Group v	Allowed edges	d_v	Discretization rule
$(\mathrm{f}, 2)$	$0 \to 2,\ 1 \to 2$	8	Two incoming edges, one operation each
$(\mathrm{f}, 3)$	$0 \to 3,\ 1 \to 3,\ 2 \to 3$	12	Two incoming edges, one operation each
$(\mathrm{f}, 4)$	$0 \to 4,\ 1 \to 4,\ 2 \to 4,\ 3 \to 4$	16	Two incoming edges, one operation each
$(\mathrm{b}, 3)$	$3 \to 0,\ 3 \to 1,\ 3 \to 2$	12	One target and its operation
$(\mathrm{b}, 4)$	$4 \to 0,\ 4 \to 1,\ 4 \to 2,\ 4 \to 3$	16	One target and its operation

Feedforward discretization first selects the operation with the highest probability on each edge and then retains the two highest-scoring incoming edges for each intermediate node. Feedback discretization selects the most probable target–operation pair for each of source nodes 3 and 4. Thus, a discrete normal cell contains six feedforward operations and two feedback operations. All operations in normal cells have stride 1 and preserve the input dimensions, whereas reduction cells contain stride-2 operations that halve the input height and width. The SNN-DARTS baseline uses the same search space.

C.2 CIFAR10/100

C.2.1 Stage 1: Architecture Search

During search, we update routing probabilities in a shallow shared-weight network with 16 initial channels. Except for the additional hyperparameters introduced by STDP, all hyperparameters are identical to those used by SNN-DARTS:

DNASNet

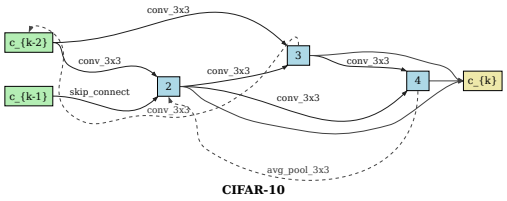
```
train:
    scheduler: cosine
    batch_size: 128
    learning_rate: 0.005
    learning_rate_min: 0.001
    momentum: 0.9
    weight_decay: 0.0003
    init_channels: 16
    layers: 8
    step: 4
    autoaugment: false
    cutout: false
    auxiliary: false
    drop_path_prob: 0
    grad_clip: 5
    node_type: BiasPLIFNode
    stdp_mu: 50.0
    stdp_mu_decay_gamma: 0.75
    alpha_init_scale: 0.001
    arch_warmup_epochs: 10
```

SNN-DARTS

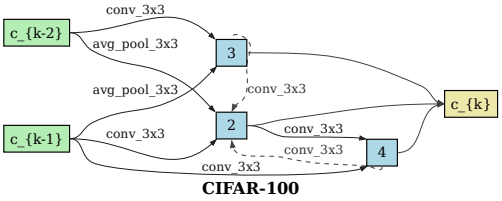
```
train:
    scheduler: cosine
    batch_size: 128
    learning_rate: 0.005
    learning_rate_min: 0.001
    momentum: 0.9
    weight_decay: 0.0003
    init_channels: 16
    layers: 8
    step: 4
    autoaugment: false
    cutout: false
    auxiliary: false
    drop_path_prob: 0
    grad_clip: 5
    node_type: BiasPLIFNode
    arch_learning_rate: 0.001
    arch_weight_decay: 0.0
    alpha_init_scale: 0.001
    arch_warmup_epochs: 10
```

As with standard DARTS, we initialize the routing probabilities in DNASNet using random perturbations on the scale of 0.001. Then, we warm up the shared weights by training the network for 10 epochs, during which the STDP utility learning step stdp_mu increases linearly from 0 to 50,

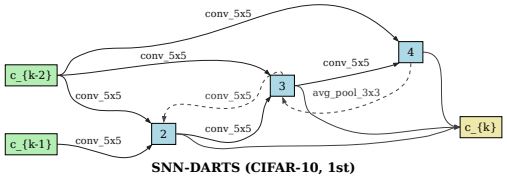
followed by exponential decay using a decay coefficient of $\gamma = 0.75$ such that convergence of the search is met. Similar SNN-DARTS step arch_learning_rate increases from 0 to 0.001 for the first 10 epochs. Figure 6 shows the cells found by DNASNet and first- and second-order SNN-DARTS.



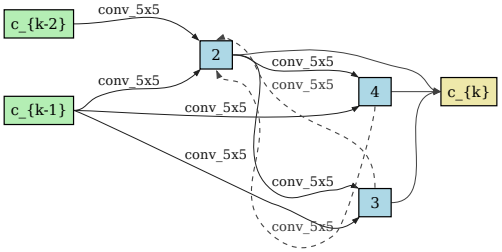
(a) DNASNet CIFAR-10



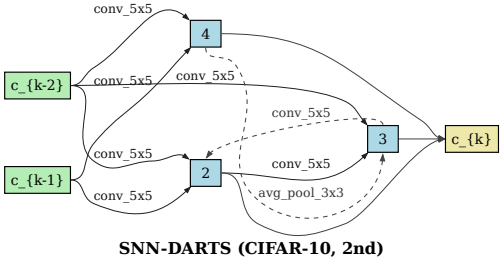
(b) DNASNet CIFAR-100



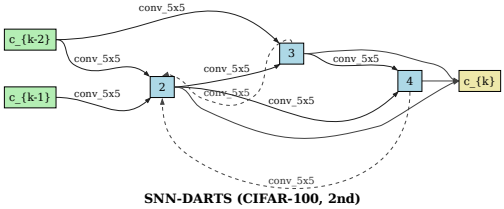
(c) SNN-DARTS(first) CIFAR-10



(d) SNN-DARTS(first) CIFAR-100



(e) SNN-DARTS(second) CIFAR-10



(f) SNN-DARTS(second) CIFAR-100

Figure 6: Cells found on CIFAR10/100. Dashed lines indicate feedback operations.

C.2.2 Stages 2 and 3: Architecture Evaluation

During evaluation, DNASNet and SNN-DARTS use identical retraining hyperparameters, and each architecture is trained for 600 epochs. The image-classification retraining configurations use AdamW (Loshchilov & Hutter, 2019).

```
train:
    act_fun: QGateGrad
    amp: false
    auxiliary: false
    batch_size: 256
    clip_grad: null
    cooldown_epochs: 10
    cut_mix: false
    cutmix_beta: 1.0
    cutmix_noise: 0.0
    cutmix_num: 1
    cutmix_prob: 0.5
    drop: 0.0
    drop_path: 0.1
    drop_path_prob: 0.2
    event_mix: false
```

```
event_size: 32
init_channels: 36
layers: 16
loss_fn: ce
lr: 0.00125
min_lr: 1.0e-05
mix_up: false
momentum: 0.9
node_type: PLIFNode
noisy_grad: 0.0
opt: adamw
opt_betas: null
opt_eps: 1.0e-08
sched: cosine
smoothing: 0.1
step: 4
tau: 2.0
threshold: 0.5
warmup_epochs: 5
warmup_lr: 1.0e-06
weight_decay: 0.01
```

Table 4 presents our extensive reproducibility results, reporting the final performance obtained with another set of random seeds for Stage 1 search. We ran search with four random seeds, but found that just 30 training epochs during the intermediate evaluation in Stage 2 were sufficient to identify and discard poorly performing architectures. This saved 50% of the cost, as we completed all 600 training epochs for only two of the 4 architectures.

Table 4: The third phase of the DNASNet evaluation consisted of two sets of randomized seeds containing one set of text-reported results.

Stage 3 Evaluation	
Set 1 (Reported)	Set 2
96.51 ± 0.10	96.48 ± 0.12

C.3 IMAGENET

Due to resource constraints, we did not search for cells independently on ImageNet. Instead, we directly transferred the best architecture found on CIFAR10 to ImageNet. This section therefore reports only the hyperparameter settings used for retraining on ImageNet.

```
act_fun: QGateGrad
amp: true
auxiliary: false
batch_size: 512
clip_grad: 1.0
cooldown_epochs: 10
cut_mix: false
cutmix_beta: 1.0
cutmix_noise: 0.0
cutmix_num: 1
cutmix_prob: 0.5
drop: 0.0
drop_path: 0.1
drop_path_prob: 0.2
event_size: 224
init_channels: 56
```

```
    layers: 16
    loss_fn: ce
    lr: 0.0012
    min_lr: 1.0e-06
    mix_up: false
    momentum: 0.9
    node_type: PLIFNode
    noisy_grad: 0.0
    opt: adamw
    opt_betas: null
    opt_eps: 1.0e-08
    sched: cosine
    smoothing: 0.1
    step: 4
    tau: 2.0
    threshold: 0.5
    warmup_epochs: 20
    warmup_lr: 1.0e-06
    weight_decay: 0.01
```

C.4 DVS-CIFAR10/DVS-GESTURE

C.4.1 STAGE 1: ARCHITECTURE SEARCH

Because DVS datasets require longer temporal sequences, we adjusted the batch size and network depth. For DVS-CIFAR-10 and DVS-Gesture, the network comprises six cells, with fixed downsampling cells at positions 2, 4, and 6. Search uses 10 time steps and a minibatch size of 64. The resulting architectures are shown in Figure 7. Due to resource constraints, SNN-DARTS was not searched on DVS-CIFAR10.The DNASNet search parameters on DVS-CIFAR10 and DVS-Gesture are configured as follows:

```
train:
    scheduler: cosine
    batch_size: 64
    learning_rate: 0.005
    learning_rate_min: 0.001
    momentum: 0.9
    weight_decay: 0.0003
    init_channels: 16
    layers: 6
    step: 10
    img_size: 48
    autoaugment: false
    cutout: false
    auxiliary: false
    drop_path_prob: 0
    grad_clip: 5
    node_type: BiasPLIFNode
    arch_learning_rate: 0.001
    arch_weight_decay: 0.0
    stdp_mu: 50.0
    alpha_init_scale: 0.001
    arch_warmup_epochs: 10
```

C.4.2 Stages 2 and 3: Architecture Evaluation

For DVS-Gesture, we limit the model's parameter count by using 10 initial channels and only four layers during 200 epochs retraining. For DVS-CIFAR10, we use the following settings and train 400 epochs.

DNASNet on DVS-Gesture

```
train:
    act_fun: QGateGrad
    amp: true
    auxiliary: false
    batch_size: 64
    clip_grad: null
    cooldown_epochs: 10
    cut_mix: false
    cutmix_beta: 1.0
    cutmix_noise: 0.0
    cutmix_num: 1
    cutmix_prob: 0.5
    drop: 0.0
    drop_path: 0.1
    drop_path_prob: 0.1
    encode: direct
    event_size: 48
    init_channels: 10
    layers: 4
    loss_fn: ce
    lr: 0.001
    min_lr: 1.0e-05
    mix_up: false
    momentum: 0.9
    node_type: PLIFNode
    noisy_grad: 0.0
    opt: adamw
    opt_betas: null
    opt_eps: 1.0e-08
    sched: cosine
    smoothing: 0.1
    step: 10
    tau: 2.0
    threshold: 0.5
    warmup_epochs: 5
    warmup_lr: 1.0e-06
    weight_decay: 0.0003
```

DNASNet on DVS-CIFAR-10

```
train:
    act_fun: QGateGrad
    amp: false
    auxiliary: false
    batch_size: 32
    clip_grad: null
    cooldown_epochs: 0
    cut_mix: false
    cutmix_beta: 1.0
    cutmix_noise: 0.0
    cutmix_num: 1
    cutmix_prob: 0.5
    drop: 0.0
    drop_path: 0.1
    drop_path_prob: 0.1
    encode: direct
    event_size: 48
    init_channels: 32
    layers: 8
    loss_fn: ce
    lr: 0.001
    min_lr: 1.0e-05
    mix_up: false
    momentum: 0.9
    node_type: PLIFNode
    noisy_grad: 0.0
    opt: adamw
    opt_betas: null
    opt_eps: 1.0e-08
    sched: cosine
    smoothing: 0.1
    step: 10
    tau: 2.0
    threshold: 0.5
    warmup_epochs: 5
    warmup_lr: 1.0e-06
    weight_decay: 0.0003
```

C.5 Experimental Results on SHD

C.5.1 Stage 1: Architecture Search

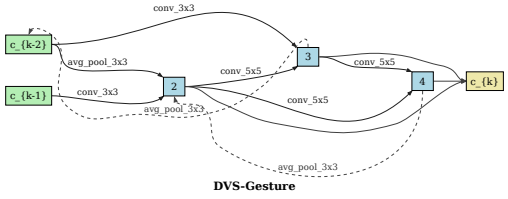
Previous work has demonstrated the benefit of learning synaptic delays on SHD (Hammouamri et al., 2024). Building on our existing SHD configuration, we construct a search space for inter-layer operations and their associated temporal delays. Specifically, the network contains four layers, each consisting of the following components:

$$\text{Linear} \rightarrow \text{BatchNorm/LayerNorm} \rightarrow \text{LIF} \rightarrow \text{Dropout}.$$

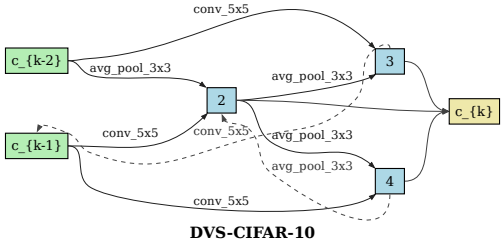
The feedback connections are fixed to $1 \rightarrow 0$, $2 \rightarrow 1$, and $3 \rightarrow 2$. The candidate operation set is

$$\mathcal{O} = \{\mathcal{I}_d, \mathcal{L}_d \mid d \in \{1, 2, 4, 8, 16, 32\}\},$$

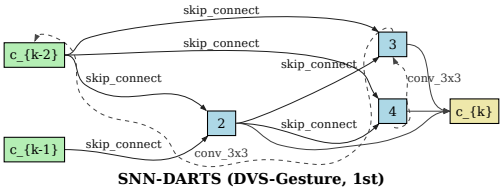
where $\mathcal{I}$ denotes an identity mapping, $\mathcal{L}$ denotes a bias-free linear layer, and d denotes the temporal delay. The search aims to identify the optimal inter-layer operations and their corresponding delays. In our search space, two parallel forward edges are used between consecutive layers. The searched architecture is shown in Figure 8.



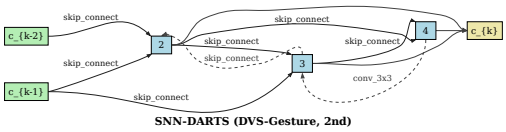
(a) DNASNet DVS-Gesture



(b) DNASNet DVS-CIFAR-10



(c) SNN-DARTS(first) DVS-Gesture



(d) SNN-DARTS(second) DVS-Gesture

Figure 7: Cells found by DNASNet and SNN-DARTS on DVS-Gesture, and by DNASNet on DVS-CIFAR-10.

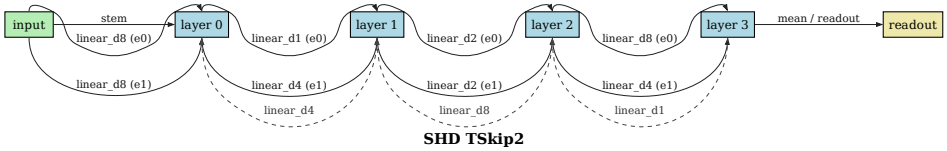


Figure 8: DNASNet architecture on SHD.

The hyperparameters used during architecture search are as follows:

```
train:
    encode: direct
    input_channels: 700
    num_classes: 20
    step: 50
    layers: 4
    init_channels: 224
    normalization: batchnorm
    batchnorm_momentum: 0.05
    dropout: 0.1
    threshold: 1.0
    neuron_type: PLIF
    act_fun: BoxcarSpike
    surrogate_half_width: 0.5
    skip_scale: 0.1
    batch_size: 128
    opt: adam
    lr: 0.01
    opt_betas: [0.9, 0.999]
    opt_eps: 1.0e-08
    weight_decay: 0.0
    loss_fn: ce
    smoothing: 0.0
    grad_clip: 1.0
    amp: false
    epochs: 80
```

```
    alpha_init_scale: 0.001
    arch_temperature: 1.0
    arch_start_epoch: 5
    stdp_lr: 0.05
    stdp_tau: 20.0
    stdp_max_delta: 0.01
    stdp_normalize: true
```

C.5.2 STAGES 2 AND 3: ARCHITECTURE EVALUATION

The SHD configurations use Adam (Kingma & Ba, 2015). The hyperparameters used for retraining are as follows:

```
train:
    encode: direct
    input_channels: 700
    num_classes: 20
    step: 50
    layers: 4
    init_channels: 224
    normalization: batchnorm
    batchnorm_momentum: 0.05
    dropout: 0.1
    threshold: 1.0
    neuron_type: PLIF
    act_fun: BoxcarSpike
    surrogate_half_width: 0.5
    skip_scale: 0.1
    batch_size: 128
    opt: adam
    lr: 0.01
    opt_betas: [0.9, 0.999]
    opt_eps: 1.0e-08
    weight_decay: 0.0
    loss_fn: ce
    smoothing: 0.0
    grad_clip: 1.0
    amp: false
    epochs: 200
    sched_patience: 5
```

D STDP HYPERPARAMETER SENSITIVITY ANALYSIS

Table 5 reports the STDP hyperparameter sensitivity experiments(mean of Kendall's τ from baseline after scanning all target values), with baseline values taken from Appendix C and the ranges listed in the table. Sensitivity is checked one hyperparameter at a time on DVS-Gesture, which provides sufficiently long temporal sequences during search while its small size allows fast experiments. The search space, data splits, shared-weight training, and discretization procedure are held fixed, and every setting follows the selection and evaluation protocol in Section C. Kendall's τ measures how the architecture found after changing one hyperparameter differs from the baseline. After each hyperparameter change, Kendall's τ relative to the baseline is calculated as follows:

$$\tau = \frac{N_c - N_d}{\sqrt{(N_c + N_d)(N_c + N_d)}}, \tag{42}$$

where the operations on each edge are ranked by their routing probabilities after search, and N_c and N_d are the numbers of concordant and discordant pairs, respectively. Intuitively, a value of τ closer to 1 indicates that the ranking of candidate operations by architecture weight is more stable after

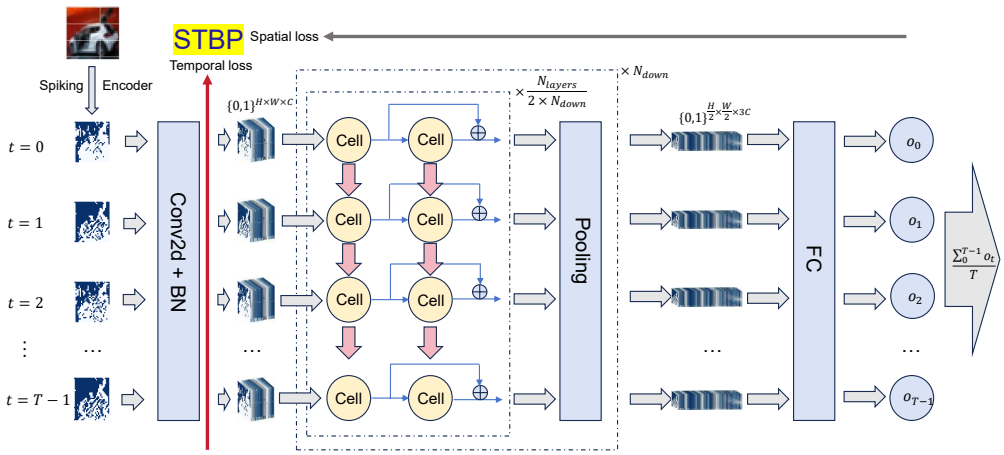


Figure 9: The DNASNet retraining network. Images are encoded as temporal inputs to the SNN; static images use direct encoding (Wu et al., 2019).

the hyperparameter change; a value closer to 0 indicates weaker rank correlation between the two settings; and a value close to -1 indicates nearly reversed rankings. Kendall's τ measures agreement in the relative ranking of architecture routing probabilities. Thus, even when $\tau = 1$, the absolute probability values need not be identical across hyperparameter settings.

Table 5: Sensitivity results for the main STDP hyperparameters.

Hyperparameter (code name)	Baseline	Planned values	Kendall's τ
STDP update strength `stdp_mu`	50	$\{10, 25, 100\}$	0.891
LTP amplitude A_+ `stdp_a_plus`	1	$\{0.5, 1, 2\}$	1.000
LTD amplitude A_- `stdp_a_minus`	1	$\{0.5, 1, 2\}$	1.000
LTP time constant τ_+ `stdp_tau_plus`	4	$\{2, 4, 8\}$	1.000
LTD time constant τ_- `stdp_tau_minus`	4	$\{2, 4, 8\}$	0.963
c_v	$2/d_v$	$\{1, 2, 4\}/d_v$	0.957
Initialization scale	0.001	$\{0.01, 0.1, 0.5\}$	0.903

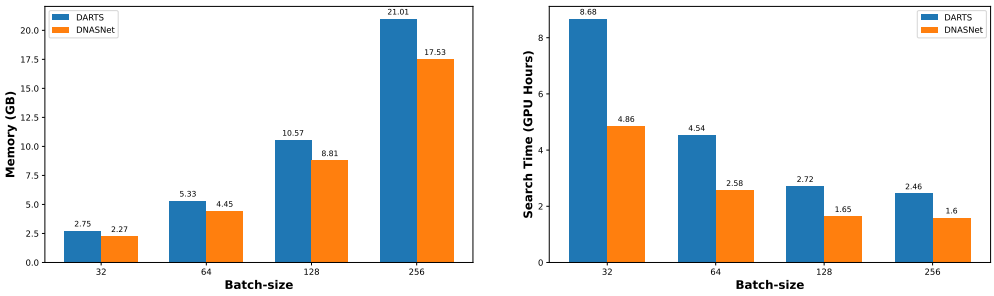
The potentiation-scale sweep uses the number of candidates d_v in each feedforward or feedback group rather than a global constant. The time constants of the event observers and candidate output neurons, candidate output thresholds, fixed routing scale, and initialization scale remain at their baseline values to avoid changing multiple factors simultaneously.

E Search Resources, Retraining Enhancements, and Dynamical Analysis

This section provides additional analysis of search costs, as well as the parameterization, ablations, and training curves for the bilinear integration in Section 4.4.

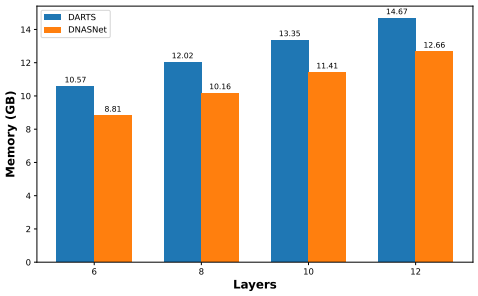
E.1 Search Efficiency

We compare DNASNet with SNN-DARTS search on CIFAR-10 using a six-layer network and $T = 4$. Figure 10 reports peak GPU memory usage and total search time for different batch sizes (a–b) and

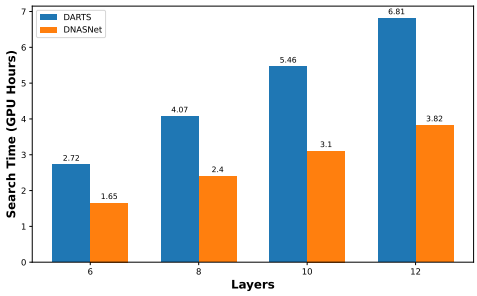


(a) Batch size vs. memory

(b) Batch size vs. search time



(c) Depth vs. memory



(d) Depth vs. search time

Figure 10: Search efficiency and scalability: peak GPU memory and search time versus batch size (a–b) and network depth (c–d), measured on an NVIDIA RTX 3090. Under these configurations, DNASNet reduces memory by approximately 20% and search time by approximately 40%–50%.

depths (c–d). Under these settings, eliminating the backward computational graph for architecture updates reduces DNASNet's memory usage and search time.

E.2 Bilinear Integration: Parameterization and Ablations

The optional retraining configuration in Section 4.4 uses

$$O^t = Y_1^t + Y_2^t + k \cdot (Y_1^t \odot Y_2^t). \tag{43}$$

At each discrete node, Y_1^t and Y_2^t are the identically shaped outputs of the two selected predecessor operations. Equation 12 retains the linear sum and models interactions through an elementwise product. Only the aggregation used during retraining is changed; the search rule and discrete topology remain fixed.

An unconstrained per-channel variable γ parameterizes the coefficient as $k = 1/(1 + \exp(-\gamma)) \in (0, 1)$ and is trained jointly with the other weights using STBP. Each output channel involved in the integration adds one learnable scalar, broadcast across the batch and spatial dimensions. The control with the module disabled fixes $k = 0$ to remove the interaction, rather than requiring the sigmoid to reach zero. Backpropagation uses

$$k = \sigma(\gamma) = \frac{1}{1 + \exp(-\gamma)} \tag{44}$$

STBP updates γ end to end. For channel c, let b index the batch and (h, w) index spatial positions. Because γ_c is shared across these positions and time steps, the chain rule sums over every broadcast dimension:

$$\frac{\partial \mathcal{L}}{\partial \gamma_c} = \sum_{t=1}^{T} \sum_{b,h,w} \frac{\partial \mathcal{L}}{\partial O_{b,c,h,w}^t} \frac{\partial O_{b,c,h,w}^t}{\partial k_c} \frac{\partial k_c}{\partial \gamma_c}. \tag{45}$$

Table 6: Ablation of the retraining enhancement with the searched topology fixed. Linear aggregation fixes $k = 0$; bilinear aggregation learns the coefficient in Equation 12.

Dataset	Bilinear integration (k)	Spike count	Accuracy (%)
CIFAR-10	× (linear sum)	610K	96.51 ± 0.10
	✓	610K	96.73 ± 0.13
CIFAR-100	× (linear sum)	589K	78.57 ± 0.11
	✓	589K	78.85 ± 0.09
ImageNet-1K	× (linear sum)	-	73.27
	✓	-	73.40

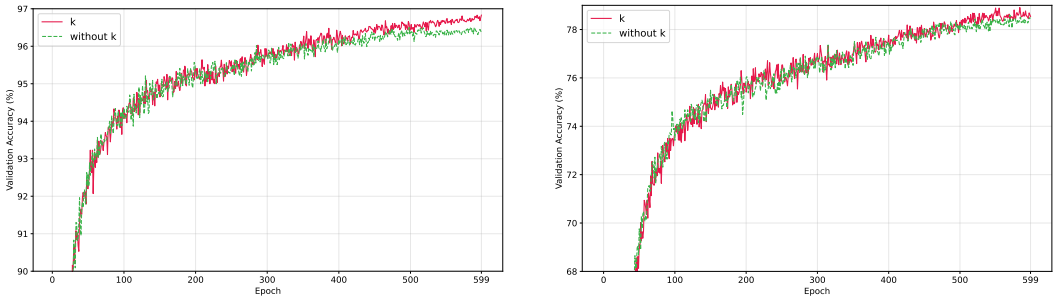


Figure 11: Validation accuracy with bilinear integration enabled and disabled under a fixed topology : CIFAR-10 (left) and CIFAR-100 (right).

The intermediate partial derivatives are

$$\frac{\partial O^t_{b,c,h,w}}{\partial k_c} = Y^t_{1,b,c,h,w} Y^t_{2,b,c,h,w}, \tag{46}$$

$$\frac{\partial k_c}{\partial \gamma_c} = \sigma(\gamma_c)\big(1 - \sigma(\gamma_c)\big) = k_c(1 - k_c). \tag{47}$$

Substitution yields the scalar gradient for each channel:

$$\frac{\partial \mathcal{L}}{\partial \gamma_c} = k_c(1 - k_c) \sum_{t=1}^{T} \sum_{b,h,w} \frac{\partial \mathcal{L}}{\partial O^t_{b,c,h,w}} Y^t_{1,b,c,h,w} Y^t_{2,b,c,h,w}. \tag{48}$$

Any averaging in the loss is already included in $\partial \mathcal{L}/\partial O^t_{b,c,h,w}$ and is not applied again. The forward pass has already computed and stored Y^t_1 and Y^t_2, so this gradient requires elementwise products and reductions. Bilinear integration thus enhances nonlinear representation capacity with little additional computation and memory.

Table 6 compares linear and bilinear aggregations where topology and other training settings remain unchanged. Accuracy improves from 96.51% to 96.73% on CIFAR-10, 78.57% to 78.85% on CIFAR-100, and 73.27% to 73.40% on ImageNet. Spike counts per sample remained at 610K and 589K for CIFAR-10 and CIFAR-100, respectively.