MAKING OF THE ANN-SNN CONTINUUM USING SPIKING THRESHOLD GATING CELLS

Anonymous authors
Paper under double-blind review

ABSTRACT

Spiking neural networks (SNNs), inspired by the brain and motivated by the potential for energy efficiency, replace real-valued activations with spikes generated by stateful neurons, but the discontinuity of spike generation complicates gradient-based training. Building on Threshold Gating (TG) [28], which expresses neural nonlinearities as affine branches selected by sigmoid gates with tunable sharpness, we introduce Spiking Threshold Gating (STG), a parameterized cell enabling a differentiable transition from stateless ANN activations to stateful spiking neurons. Unlike prior ANN-to-SNN conversion methods, STG allows ANN pretraining and subsequent hardening within the same cell, using exact gradients of the smooth cell during training rather than surrogate gradients. We then introduce a curriculum that progressively sharpens the cell's gates during training, achieving state-of-the-art results on several event-based datasets. We additionally propose a novel input-routed parameterization that separates membrane-driven firing from input-dependent event payloads, further improving performance through flexible, multi-branch emissions.

1 INTRODUCTION

ANNs spend much of their energy moving dense activations and weights, whereas SNNs communicate spiking events, allowing neuromorphic "brainlike" hardware to skip both the arithmetic and data movement when a neuron is *inactive* [18, 20, 6, 25]. This offers potential to reduce energy consumption, a growing concern as AI systems become more computationally demanding [7].

The distinction between a conventional ANN activation and a spiking neuron can be understood through the basic leaky integrate-and-fire (LIF) model. A LIF neuron *integrates* inputs in a membrane state; *leakage* gradually reduces the influence of earlier inputs. It *fires* when the membrane state crosses a threshold, after which a *reset* reduces the state before accumulation continues [7]. Figure 1 visualizes this difference by comparing stateless ANN activations and stateful neurons responding to the same input.

In a LIF neuron, integration and leakage are continuous operations; however, discontinuity enters through the hard threshold decision that generates a spike and triggers a reset. Spiking reduces arithmetic and data movement on event-driven hardware [6, 25], but the hard spike function has a zero derivative almost everywhere and an undefined derivative at the threshold, making gradient-based training difficult. In contrast, conventional ANN activations are continuous and differentiable almost everywhere, acting on the current input without carrying state. These differences can be summarized along two axes—statefulness and output type (graded or binary)—as shown in Figure 1.

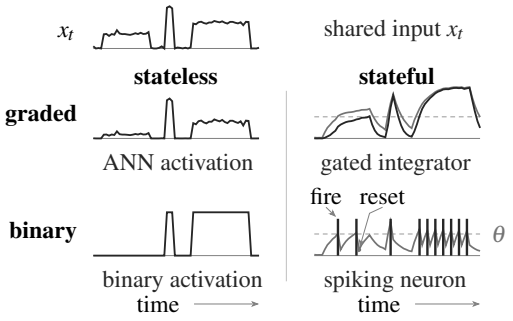

Figure 1: From activations to spikes: output and memory. Shared-input responses by output type and statefulness. Black: output; grey: state; dashed: threshold. All four are STG configurations.

Among the approaches used to address this difficulty, two are particularly common: surrogate-gradient training and ANN-to-SNN conversion. Surrogate-gradient methods retain hard spikes in the

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.

forward pass and substitute a smooth derivative during backpropagation [23, 37, 31]. ANN-to-SNN conversion trains an ANN first and transfers its learned computation to a spiking network [4, 26, 30]. However, surrogate-gradient learning introduces a mismatch between the forward computation and the derivative used for training, while ANN-to-SNN conversion introduces a mismatch between ANN training and spiking inference dynamics. A potential solution lies in using exact gradients while training towards *spikyness*.

Threshold Gating (TG) expresses neural nonlinearity as gated affine branches [28]. For example, $\mathrm{ReLU}(x) = x\mathbf{1}[x > 0]$ selects between the input and zero, placing the nonlinearity in the gate instead of an explicit non-linear activation. TG then replaces the indicator with $g(x) = \sigma(\tau_o(x - \theta_{TG}))$ making the gating smooth, with threshold θ_{TG} and sharpness τ_o. The indicator (hard) function becomes the limit of the smooth gate as $\tau_o \to \infty$.

The *useful property* of TG is that the soft gate remains differentiable at every finite τ_o, even as it gets larger. Training can therefore use exact gradients while moving toward a hard threshold decision. Another *key observation* is that the physical substrate or hardware does not require the mathematical limit $\tau_o \to \infty$: at finite output precision, a sufficiently sharp gate is indistinguishable from a binary decision outside a narrow transition band (Section 4). All that is left to obtain a basic spiking cell is to add a leaky membrane state that accumulates input, controls event emission, and resets after firing. Building on this construction, we introduce Spiking Threshold Gating (STG), a broader parameterized family encompassing ANN activations and adaptive or recurrent spiking dynamics.

Contributions. **(1) STG cell.** STG combines membrane and adaptation dynamics with gated affine output branches in one parameterized cell. Its parameters specify how the state evolves and what the cell emits (Section 2 and Table 1). **(2) Continuum.** The same cell connects ANN activations to canonical spiking neurons through its parameter settings. Finite gate sharpness supports exact-gradient training, while a sharpness curriculum moves the cell toward spiking emission (Sections 3 and 5; proofs in Appendices D and F). **(3) Routed payloads.** Routing lets the membrane control branch selection while the input determines the event payload. This separates firing from the value emitted and gives the payload a direct gradient path around the membrane recurrence (Section 6.3 and Theorem 3).

2 Spiking Threshold Gating (STG)

Threshold Gating [28] expresses a neural nonlinearity as a weighted sum of K affine branches:

$$\mathrm{TG}(x) = \sum_{k=1}^{K} \pi_k(x)\,(s_k x + c_k), \tag{1}$$

where s_k and c_k are the branch slope and offset, and the soft branch weights satisfy $\pi_k(x) \geq 0$ and $\sum_k \pi_k(x) = 1$. These weights are constructed from sigmoid threshold gates $\sigma(\tau_o(x - \theta_{TG}))$, with $\tau_o > 0$ controlling gate sharpness (emission sharpness in STG).

Two properties make this a useful starting point. First, the branch weights remain non-negative and sum to one, allowing a smooth transition from blending affine branches to selecting among them as sharpness increases. Second, every finite sharpness τ_o gives a differentiable function with exact gradients. We extend TG with stored states, gated state updates, and an emission stage. The states are updated and the output is generated using TG (Figure 2). In the following, we describe the state updates and emission mechanism of STG.

State-dependent gating. In TG, each gate compares the current input x with a learned threshold $\theta_{TG}^{(k)}$. To introduce memory, each STG cell maintains two states: a membrane potential h_t that accumulates input, and an adaptation state r_t that tracks activity to adjust the firing threshold. Beyond retaining past inputs, these states can also change how readily the cell fires. Motivated by voltage-dependent thresholds [21] and adaptive firing thresholds [2], we let both states shift the TG threshold. Before processing the input x_t, the states are h_{t-1} and r_{t-1}. These states control the gates used to compute h_t and r_t. The resulting threshold is $\theta_{\mathrm{STG},t-1}^{(1)} = \theta_{TG}^{(1)} + \kappa h_{t-1} + \rho r_{t-1}$. Here $\theta_{TG}^{(1)}$ is the first learned TG threshold, which is fixed across timesteps. The learned coefficients κ and ρ scale the membrane and adaptation contributions, respectively. Setting both coefficients to zero leaves a fixed

threshold. Subtracting this STG threshold from h_{t-1} gives the margin z_t:

$$z_t = h_{t-1} - \theta^{(1)}_{\mathrm{STG},t-1} = (1-\kappa)h_{t-1} - \rho r_{t-1} - \theta^{(1)}_{\mathrm{TG}}, \tag{2}$$

We apply TG's sigmoid to this margin to obtain the *state-update gates* for the membrane and adaptation:

$$g_{h,t} = \sigma(\tau_{\mathrm{h}} z_t), \qquad g_{r,t} = \sigma(\tau_{\mathrm{r}} z_t), \tag{3}$$

The learned sharpnesses τ_{h} and τ_{r} control how abruptly the two state updates switch branches. They are independent of the emission sharpness τ_{o}. Both gates are differentiable at any finite sharpness.

Membrane update. Using $g_{h,t}$, we update the membrane by blending two learned affine branches:

$$h_t = (1-g_{h,t})\big(s_{h,1}h_{t-1} + w_{h,1}x_t + c_{h,1}\big) + g_{h,t}\big(s_{h,2}h_{t-1} + w_{h,2}x_t + c_{h,2}\big). \tag{4}$$

Here $s_{h,j}$ controls retained membrane activity, $w_{h,j}$ scales the input, and $c_{h,j}$ adds an offset.

Adaptation update. We update the adaptation state in the same way, using $g_{r,t}$:

$$r_t = (1-g_{r,t})\big(s_{r,1}r_{t-1} + w_{r,1}x_t + c_{r,1}\big) + g_{r,t}\big(s_{r,2}r_{t-1} + w_{r,2}x_t + c_{r,2}\big). \tag{5}$$

The coefficients have analogous roles; $s_{r,j} \in (0,1)$ closer to one retains adaptation longer within that branch. The updated states then set the STG thresholds:

$$\theta^{(k)}_{\mathrm{STG},t} = \theta^{(k)}_{\mathrm{TG}} + \kappa h_t + \rho r_t, \qquad \theta^{(1)}_{\mathrm{TG}} < \cdots < \theta^{(K-1)}_{\mathrm{TG}}. \tag{6}$$

Each STG threshold is recomputed from its learned TG threshold and the current states; it does not accumulate shifts from previous thresholds. Both state-dependent terms shift all thresholds equally, preserving their ordering and spacing. Positive κ or ρ raises thresholds as the corresponding state increases, while negative values lower them. Raising a threshold makes it harder to cross for a given gating input, providing activity-dependent feedback.

Output generation (emission). With the states updated, the cell produces its output for the current timestep; we call this stage *emission*. Stateless TG uses the same input for gating and affine branches. STG separates these roles: the membrane can control firing, while the signal supplied to the output branches (the *payload*) can come from the membrane, the input, or a blend of both. We define $X_t^{\mathrm{gate}} = \beta_{\mathrm{g}} h_t + (1-\beta_{\mathrm{g}})x_t$ and $X_t^{\mathrm{pay}} = \beta_{\mathrm{p}} h_t + (1-\beta_{\mathrm{p}})x_t$, where $\beta_{\mathrm{g}}, \beta_{\mathrm{p}} \in [0,1]$ control how much each signal depends on the membrane rather than the current input.

With β_{g}=1 and β_{p}=0, the membrane controls branch selection while the current input supplies the payload (STG$_{\mathrm{ir}}$; Section 6.3). With β_{g}=β_{p}=1, the membrane supplies both signals.

For emission, $\ell \in \{0,1\}$ selects which timestep's thresholds to use. By default, $\ell = 0$, so $t-\ell = t$: we use $\theta^{(k)}_{\mathrm{STG},t}$, computed from the updated states h_t, r_t. Optionally, $\ell = 1$, so $t-\ell = t-1$: we use $\theta^{(k)}_{\mathrm{STG},t-1}$, computed from the previous states h_{t-1}, r_{t-1}. In both cases, the gating signal is the current X_t^{gate}. To determine how strongly each output branch contributes, we first compare the gating signal with each threshold. The index $k = 1, \ldots, K-1$ labels the thresholds in increasing order. The sigmoid gate $q_t^{(k)} = \sigma\big(\tau_{\mathrm{o}}(X_t^{\mathrm{gate}} - \theta^{(k)}_{\mathrm{STG},t-\ell})\big)$ smoothly indicates whether the gating signal exceeds the corresponding threshold value $\theta^{(k)}_{\mathrm{STG},t-\ell}$, approaching zero below it and one above it. Neighbouring threshold gates define the output-branch weights: $\pi_{k,t} = q_t^{(k-1)} - q_t^{(k)}$. We set $q_t^{(0)} = 1$ and $q_t^{(K)} = 0$ so this rule also covers the first and last branches, whose weights are $1 - q_t^{(1)}$ and $q_t^{(K-1)}$, respectively. The output is the weighted sum of the branch values, retaining the TG form in (1):

$$\boxed{\mathrm{STG}(x_t) = \sum_{k=1}^{K} \pi_{k,t}\big(s_k X_t^{\mathrm{pay}} + c_k\big).} \tag{7}$$

In (7), K counts the output branches $s_k X_t^{\mathrm{pay}} + c_k$. For example, $K = 3$ gives three output branches, while the membrane and adaptation updates in (4) and (5) still use two branches each. Table 1 summarizes three output and routing configurations.

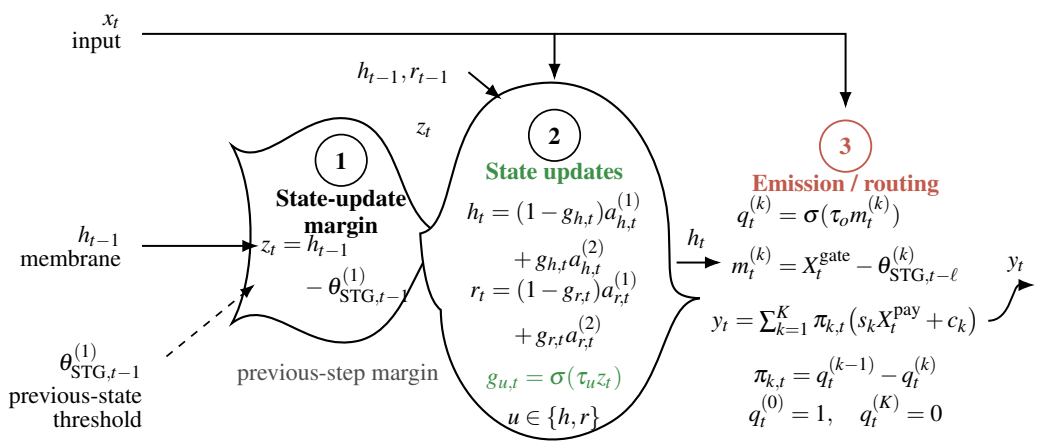

Figure 2: The cell in three stages. Previous-state margin, membrane and adaptation updates, and emission $y_t = \mathrm{STG}(x_t)$. The states shift the thresholds through $\kappa h + \rho r$; emission thresholds use current ($\ell = 0$) or previous ($\ell = 1$) states. The output branches give binary or graded emissions (Table 1).

Why the cell remains differentiable. Hard threshold decisions can introduce discontinuities in both spike emission and the associated membrane reset. In STG, sigmoid gates blend affine branches in both the state updates and emission, so their composition remains differentiable for every finite τ_h, τ_r, and τ_o. Increasing sharpness narrows these transitions without introducing a hard switch at any finite value; gradients remain exact, though they can become small away from thresholds. Hard switching arises only in the infinite-sharpness limit or through an explicit discrete operation.

Table 1: Three STG configurations, all with $\beta_g = 1$. Binary denotes sharp-limit $\{0, C\}$ output, away from threshold ties; $C = 1$ gives unit spikes. Finite sharpness blends the branches.

Configuration	Output branches	Routing
Binary STG	$K = 2$; fix $s_1 = s_2 = 0$, $c_1 = 0$, $c_2 = C$. One silent branch and one constant branch.	Payload routing has no effect.
Membrane-graded STG	$K = 2$ or 3; learn s_k, c_k, with no branch forced to zero.	$\beta_p = 1$: membrane payload.
Input-graded STG ($\mathrm{STG_{ir}}$)	$K = 2$ or 3; learn s_k, c_k, with no branch forced to zero.	$\beta_p = 0$: input payload.

What the emission parameters control. Increasing τ_o narrows the transition between branches. The slopes s_k scale the selected signal, while c_k sets each branch's baseline. Nonzero slopes preserve amplitude information but require graded output values rather than LIF's binary events. With s_k=0, the sharp-limit outputs are fixed levels c_k; a branch fixed at (s_k, c_k)=$(0, 0)$ is a silent, or null, branch. Increasing K adds output levels or affine responses at the cost of more parameters. Recovering a specific neuron model also requires its state-update and reset settings (Table 2).

3 The ANN–SNN Continuum

With our STG cell, ANNs with typical activation functions and canonical spiking neurons become part of a differentiable continuum. At any finite sharpness of its gates, the cell is differentiable and supports exact gradients. At the smooth end, disconnecting the states gives stateless TG, including SiLU and an approximation to GELU. At the sharp (spiking) end, binary emission with the appropriate membrane and reset dynamics recovers canonical spiking neurons in the hard-gate limit.

Within the spiking configurations, the output can be binary, multi-level, or graded. Zero branch slopes give fixed output levels: two levels $\{0, C\}$ give binary spikes, while additional branches can provide multiple spike amplitudes. Nonzero slopes retain amplitude information within each branch and give graded output. For graded output, routing is a separate choice: the membrane can determine when to emit while either the membrane or the current input supplies the emitted value (Section 2, Table 1); Section 6.3 examines the resulting gradient paths.

Table 2: STG settings that recover ANN activations and spiking neurons. Insert the state branches into (4)–(5) and the output coefficients into (7). Appendix D.1 gives the substitutions explicitly.

Activation / neuron	τ_o	(β_g, β_p)	$(a^{(1)}_{h,t}, a^{(2)}_{h,t})$	$(a^{(1)}_{r,t}, a^{(2)}_{r,t})$	(s_k, c_k)
SiLU	1	$(0,0)$	not read	not read	$(0,0),(1,0)$
GELU (approx.)	1.702	$(0,0)$	not read	not read	$(0,0),(1,0)$
ReLU	$\to\infty$	$(0,0)$	not read	not read	$(0,0),(1,0)$
LIF	$\to\infty$	$(1,1)$	$(\lambda h_{t-1}+U_t, U_t)$	$(0,0)$	$(0,0),(0,1)$
Spike-adaptive LIF	$\to\infty$	$(1,1)$	$(\lambda h_{t-1}+U_t, U_t)$	$(R_t, R_t+1-\gamma)$	$(0,0),(0,1)$
Recurrent spike-adaptive LIF	$\to\infty$	$(1,1)$	$(\lambda h_{t-1}+U^{\mathrm{rec}}_t, U^{\mathrm{rec}}_t)$	$(R_t, R_t+1-\gamma)$	$(0,0),(0,1)$
Multi-level STG	$\to\infty$	$(1,1)$	$(\lambda h_{t-1}+U_t, U_t)$	$(0,0)$	$(0, c_k)$
Graded STG	$\to\infty$	$(1,\beta_p)$	$(\lambda h_{t-1}+U_t, U_t)$	$(0,0)$	(s_k, c_k)

$U_t = (1-\lambda)x_t$, $R_t = \gamma r_{t-1}$, and $U^{\mathrm{rec}}_t = (1-\lambda)(x_t + \mathbf{V}S_{t-1})$. All rows use $\kappa = 0$, $\ell = 0$; ANN rows also use $\rho = 0$ and $\theta_{\mathrm{TG}} = 0$. Spiking rows use hard state-update gates and hard reset to zero. Only the adaptive rows use $\rho \neq 0$, giving $\theta^{(1)}_{\mathrm{STG},t} = V_{\mathrm{th}} + \rho r_t$; other spiking thresholds are constant. Binary rows use $K = 2$; multi-level rows allow $K > 2$ distinct c_k, including zero. For graded output, $\beta_p = 1$ reads the membrane and $\beta_p = 0$ reads the input. Other reset choices are given in (11).

Sharpness alone does not make a cell an SNN: the state and output settings also matter, as the ReLU limit illustrates. A compatible ANN can initialize a smooth STG network, whose gates can then be sharpened during training. Exact activation and neuron equivalences are established in Appendix D; differentiability at any finite sharpness is established in Appendix F. The next section explains when finite sharpness is sufficient for spiking emission at the resolution used to represent the output.

4 Spiking Emission at Finite Sharpness

How large must τ_o be for emission to behave like a spike? A finite sigmoid never becomes an exact step, but its output can be indistinguishable from a hard gate at the numerical or physical resolution of the emitted signal. The required sharpness depends on that resolution and on how far the firing signal lies from the threshold.

Let $0 < \varepsilon < 1/2$ be the smallest distinguishable departure of a normalized gate value from zero or one. For $q(m) = \sigma(\tau_o m)$, where m is the emission margin, intermediate gate values remain resolvable only within

$$\mathscr{T}_\varepsilon(\tau_o) = \{m : \varepsilon < q(m) < 1-\varepsilon\} = \left\{ m : |m| < \frac{1}{\tau_o} \log \frac{1-\varepsilon}{\varepsilon} \right\}. \tag{8}$$

Outside this interval, the gate is indistinguishable from zero or one at resolution ε. Thus, for margins satisfying $|m| \geq d > 0$, it suffices to choose $\tau_o \geq d^{-1} \log((1-\varepsilon)/\varepsilon)$. For example, an 8-bit uniform representation of the gate on $[0,1]$ with round-to-nearest has $\varepsilon = 1/510$; its transition interval has half-width approximately $6.23/\tau_o$ in margin units. There is consequently no universal sharpness that makes every emission spiking: margins near a threshold can remain intermediate even at large τ_o.

For binary branches 0 and C, the criterion applies to the normalized emission y/C. Multiple fixed levels give multi-level output, whereas nonzero slopes preserve a graded payload even with hard gates. A comparator enforces hard branch selection; a quantizer instead rounds the emitted value to a finite set of levels. For an output quantizer with a zero level and uniform spacing Δ, values with $|y| < \Delta/2$ round to zero. The event rate—the fraction of neuron-timesteps with nonzero output—therefore depends on this zero interval and the output distribution, not just the bit width.

Only emission passes between layers, so internal state-update gates can remain smooth when the output is discretized. Our quantized experiments retain floating-point states and specify emission and

synaptic precision separately. The criterion above concerns static gate resolution; analog propagation and settling time are separate quantities.

5 Training on the Continuum

A sharpness curriculum trains STG from smoother gates toward sharper emission. We use two forms: a *scheduled curriculum* prescribes how sharpness increases, while a *learned curriculum* optimizes sharpness under a gradually increasing penalty. Both implement the same smooth-to-sharp training principle. *Fixed-sharpness training* holds sharpness constant and serves as a comparison. All three use exact gradients through the smooth cell. Emission quantization and fine-tuning are addressed separately below.

Before training, we choose the state dynamics and output configuration (Table 1). Binary STG fixes the branch slopes at zero and includes a silent branch; graded STG learns the slopes and offsets. These choices determine what the cell can emit. The training procedure determines how sharply it selects between those branches. In all three procedures, we write emission sharpness as $\tau_o(\zeta) = \tau_o^{\min} + (\tau_o^{\max} - \tau_o^{\min})\zeta$, with $\zeta \in (0,1)$. The membrane and adaptation sharpnesses are learned separately.

Scheduled curriculum. A smooth increase in ζ is prescribed over training epochs e:

$$\zeta(e) = \zeta_0 + \frac{\zeta_1 - \zeta_0}{1 + \exp(-\nu(e - e_0))}. \tag{9}$$

Here ζ_0 and ζ_1 are the limiting values, e_0 is the midpoint, and $\nu > 0$ sets the rate of increase. Early training blends branches more smoothly; later training approaches hard selection. This is the scheduled form of the sharpness curriculum.

Learned curriculum. ζ_{raw} is optimized jointly with the network, using $\zeta = \sigma(\zeta_{\text{raw}})$. A gradually increasing penalty encourages sharper emission:

$$\mathscr{L} = \mathscr{L}_{\text{task}} + \alpha(n)(1-\zeta)^2, \qquad \alpha(n) = \alpha_{\max}\sigma(\nu_\alpha(n - f_0 N)). \tag{10}$$

Here n is the training step, N the total number of steps, $f_0 \in (0,1)$ the midpoint fraction, and $\nu_\alpha > 0$ the penalty growth rate; the midpoint occurs at step $f_0 N$. Unlike the prescribed schedule, this procedure lets the task loss influence the sharpness trajectory; it need not increase monotonically at every optimization step.

Fixed sharpness. ζ is held constant throughout training while the remaining trainable parameters are optimized. Choosing a low or high value gives a smooth or sharp gate from the outset. This comparison tests whether changing sharpness during training improves on training the same cell at a constant sharpness. Dataset-specific settings are given in Appendix B.

Quantization and fine-tuning. Hard branch selection yields discrete levels for constant payloads, but graded payloads can still vary continuously, including with input routing. Such emissions require quantization or another explicit discretization to obtain a prescribed finite alphabet (Section 4). Quantization rounds outputs to that alphabet; *quantized fine-tuning* trains with the quantizers active to recover accuracy. Discretization itself does not require further training.

Emission sharpness is fixed, quantizer scales are calibrated, and fine-tuning proceeds at the target precision. Where synaptic quantization is used, weights and activations use eight bits (W8A8); cell states remain floating point. Rounding uses a straight-through estimator (STE), so the exact-gradient property of smooth cell training does not extend to that operation. Floating-point reference accuracies are obtained before this stage. SHD, SSC, and CIFAR10-DVS use quantized fine-tuning; MSWC-FSCIL uses emission quantization during evaluation. Quantized accuracy and event rate in Table 4 are measured on the same model with the quantizers active.

6 Evaluation and Discussion

The evaluation covers two auditory benchmarks, SHD and SSC [5]; two visual benchmarks, CIFAR10-DVS [14] and DVS-Gesture [1]; and few-shot class-incremental learning on MSWC [19, 35]. We

Table 3: Test accuracy (%). Our results give means ± standard deviations across three seeds; † marks one seed. Bold values are the highest listed. E4 and E8 use 4- and 8-bit emissions with W8A8; MSWC-FSCIL quantizes emissions only. States remain FP32, and FP32 rows are unquantized. SHD and DVS-Gesture report peak accuracy; MSWC-FSCIL reports the ten-session mean.

Benchmark	Method	Network / approach	Training	Accuracy
SHD	DCLS-Delays [10]	Learnable synaptic delays	Surrogate	95.07
	STG (ours)	DCLS-Delays, 2×256, FP32	Exact, curriculum	96.04 ± 0.60
	STG (ours)	DCLS-Delays, 2×384, E4	Curric. + QAT	$\mathbf{96.21 \pm 0.69}$
SSC	DCLS-Delays [10]	Learnable synaptic delays	Surrogate	80.69
	SpikCommander [33]	Spiking transformer, $T=100$	Surrogate	83.49
	STG (ours)	S4D, input-graded E8	Curric. + QAT	$\mathbf{83.91 \pm 0.37}$
CIFAR10-DVS	STAA-SNN [38]	Spatio-temporal attention, VGG-13	Surrogate	82.10
	QKFormer [39]	Spiking transformer	Surrogate	84.00
	STG (ours)	ConvNeXt-Small (ImageNet), input-graded E4	Curric. + QAT	$\mathbf{93.47 \pm 0.32}$
DVS-Gesture	DTS-SNN, our run	Same network and budget	Surrogate	94.44†
	STG (ours)	DTS-SNN, binary-limit FP32	Exact, curriculum	**95.83**†
MSWC-FSCIL	RadLIF [35]	NeuroBench recurrent SNN	Surrogate	75.27
	STG (ours)	NeuroBench recurrent SNN, E4	Curriculum	**83.32**†

use DCLS-Delays for SHD [10], a five-layer S4D network for SSC [8], and ConvNeXt-Small with $T = 16$ for CIFAR10-DVS [16]. The remaining backbones are DTS-SNN for DVS-Gesture [36] and the NeuroBench recurrent SNN for MSWC-FSCIL [35]. Table 3 compares accuracy with published spiking models, and Table 4 lists the configuration, size, and quantized results of each STG model; Appendix B specifies the training and evaluation settings.

6.1 Accuracy across Cell Configurations

Table 3 compares STG with published and reproduced baselines across tasks and output configurations. SHD uses three graded branches; SSC and CIFAR10-DVS use two with input routing. DVS-Gesture and MSWC-FSCIL use two constant branches, binary in the sharp limit (Table 1). Quantized results have finite emission alphabets; FP32 references retain unquantized, finite-sharpness outputs.

On SHD, unquantized STG gives $96.04 \pm 0.60\%$ at width 256, versus $94.72 \pm 0.36\%$ for our LIF reference, with 3.8% more parameters and the same augmentation. Width 384 gives $97.08 \pm 0.27\%$ unquantized and $96.21 \pm 0.69\%$ with E4 emission, at approximately twice the reference model's size. These are peak test accuracies; Appendix B also reports averages over the final training epochs.

On SSC, SpikCommander reports 83.49% at $T = 100$ [33]. Quantized STG gives 83.91% with 8-bit emission (83.03% with 4-bit). The SSC E8 result exceeds the $T = 100$ reference by 0.42 points; this comparison is restricted to the approximately 100-step setting. The unquantized SSC reference is $84.46 \pm 0.23\%$. Architectures and training procedures differ across these comparisons.

On CIFAR10-DVS, our pretrained ConvNeXt-Small configuration reaches $94.10 \pm 0.17\%$ with unquantized input-graded emissions and finite-sharpness gates. Quantized fine-tuning gives $93.47 \pm 0.32\%$ with 4-bit emissions (Table 4). For context, QKFormer reports 84.0% and I2E reports 92.5% with synthetic-event pretraining [39, 17]. Differences in pretraining, model size, emission format, and data split prevent attributing this gap to the cell alone.

On MSWC-FSCIL, 4-bit emission gives 83.32% (one seed), 8.05 points above the published NeuroBench RadLIF reference [35]. The unquantized three-seed reference is $84.12 \pm 0.13\%$. On DVS-Gesture, the comparison uses the same DTS-SNN backbone and 200-epoch budget. STG improves peak accuracy by 1.39 points and the mean over the final five epochs by 3.26 points in this single-seed comparison. DTS-SNN reports 96.06% [36], whereas our reproduction obtained 94.44%.

Table 4: STG configurations and quantized results. Three seeds are used unless marked †, which indicates one seed. Accuracies and event rates are means ± standard deviations; dashes indicate unreported results. τ_o^{max} bounds emission sharpness, and T counts timesteps. Emissions use 4 bits (8 for E8) and one scale per tensor. Synapses use W8A8 except on MSWC-FSCIL. Event rates count nonzero quantized outputs.

							Quantized	
Benchmark	**Network**	**Output**	K	τ_o^{max}	T	**Params**	**Acc. (%)**	**Events (%)**
SHD	DCLS-Delays, 2×256	graded	3	10	≈100	332.8K	—	—
	DCLS-Delays, 2×384	graded	3	10	≈100	646.7K	96.21 ± 0.69	13.31 ± 2.21
SSC	S4D, 5×512	input-graded	2	50	≈100	1.53M	83.03 ± 0.20	17.68 ± 1.34
	S4D, 5×512 (E8)	input-graded	2	50	≈100	1.53M	83.91 ± 0.37	23.66 ± 1.95
CIFAR10-DVS	ConvNeXt-Small	input-graded	2	10	16	50.6M	93.47 ± 0.32	8.43 ± 0.17
DVS-Gesture	DTS-SNN	binary	2	10	300	1.27M	—	—
MSWC-FSCIL	NeuroBench RSNN	binary	2	10	201	—	83.32†	1.56†

6.2 Effect of the Sharpness Curriculum

A matched SSC comparison tests smooth-to-sharp training against hard forward gates with surrogate gradients, holding the backbone, cell parameters, and other training settings fixed. The curriculum gives $84.40 \pm 0.24\%$, compared with $83.80 \pm 0.23\%$ for the hard-gate baseline (three seeds per setting). The 0.60-point improvement supports the combined use of a smooth forward gate and gradual sharpening; this comparison alone does not separate their individual effects. Both results precede quantized fine-tuning.

Appendix G gives the rationale for this order of training. For a fixed input distribution, increasing sharpness concentrates gate derivatives near thresholds and increases their variance asymptotically. Smoother early gates provide exact derivatives over a wider range of margins, helping the network learn branch assignments. Once margins are separated from thresholds, further sharpening changes the emitted values exponentially little. This supports later sharpening of an existing solution, without assuming that large gate-derivative variance is always harmless. The benefit remains configuration dependent: a three-seed ResNet-20 experiment on DVS-Gesture favors fixed sharpness over annealing (Appendix C).

6.3 Graded Output and Input Routing

Input routing separates when a cell fires from what it emits: $\beta_g = 1$ lets the membrane select the branch, while $\beta_p = 0$ lets the input supply its value. TG observes that nonzero branch slopes s_k provide a direct input-gradient term in addition to the gate derivative [28]. Input routing carries this path directly from the payload to the current input, bypassing the membrane recurrence (Appendix E). The SSC and CIFAR10-DVS models in Table 4 use this configuration.

For binary emission, $s_k = 0$ removes this path and makes β_p irrelevant. Routing chooses between fixed and signal-dependent payloads; it does not determine statefulness. Routing and branch count vary by task, so cross-dataset results do not isolate their effects. Ratios of learned slopes to offsets also depend on payload scale and do not establish a universal routing rule.

6.4 Quantized Emission and Event Rate

Table 4 gives precision and event rates for the quantized models in Table 3, plus the SSC E4 alternative. Relative to the unquantized reference models, E4 reduces accuracy by 0.87, 1.43, and 0.63 points on SHD, SSC, and CIFAR10-DVS; SSC E8 reduces it by 0.55 points. SSC E8 retains more accuracy at 23.66% events, versus 17.68% for E4. Events count nonzero quantized emissions, not nonzero sigmoid gates.

A quantizer can map small graded outputs to zero without a fixed silent branch. Even with a silent branch, finite-sharpness gates require a discrete interface for exact silence. MSWC-FSCIL quantizes emissions only, without W8A8.

Event rates depend on the zero interval and scale granularity as well as bit width: reducing precision at fixed scale widens a uniform quantizer's zero interval. Event rate times payload bits estimates

payload traffic per neuron-timestep, excluding addresses, routing, and state updates; it does not measure hardware energy or latency.

7 Related Work

STG cell. UltraLIF is a close precedent: it trains smooth membrane dynamics and sigmoid emissions with exact gradients and a learnable temperature [22]. Its log-sum-exp update approaches max-plus integration as temperature decreases. STG instead selects between affine state-update branches, whose settings recover leaky integration, reset, and adaptation (Table 2). This makes the recovered dynamics explicit while allowing their state and output settings to vary separately. The distinction is the cell parameterization, not differentiability alone. Standard ultradiscretization requires a subtraction-free expression in positive original variables; UltraDLIF rewrites its diffusion update with positive, equal weights to meet this requirement. STG's affine branches use signed coefficients directly, without that transformation.

Continuum and training. Surrogate-gradient methods differentiate hard spikes using substitute derivatives [23, 37, 31], whereas EventProp and Eventax obtain gradients through event timing [34, 24, 13]. STG differentiates its smooth forward computation at finite sharpness, as does UltraLIF. The STG continuum specifies concrete parameter settings for ANN activations and canonical spiking configurations (Table 2). This lets compatible ANN weights initialize the cell that is subsequently sharpened, connecting pretraining and spiking inference without a separate neuron formulation as in conventional ANN-to-SNN conversion [4, 26, 3]. Quantized fine-tuning still uses STE for rounding.

Routed payloads. Graded spikes already appear in prior work: Real Spike learns real-valued spikes during training and uses binary spikes at inference [9]; Loihi 2 supports integer-valued payloads [25, 32]. In Orchard et al.'s resonate-and-fire model, a state crossing determines event timing and the resonator magnitude supplies its payload [25]. STG makes the sources of these two signals separate parameters: the membrane can select a branch while the current input supplies its value, yielding the direct payload gradient path in Theorem 3. The cochlea offers a biological analogy for separating graded processing from spike generation: graded hair-cell potentials drive all-or-none auditory-nerve spikes [27]. That analogy concerns separate processing stages; STG's input-dependent event amplitudes are a modeling choice, not a claim about auditory-spike amplitudes.

8 Limitations

As a training procedure, the only limitation could be the sharpening of τ and the most appropriate curriculum and schedule for it. Even though in our experiments the curriculum-based training remained fairly robust, more experiments can be done to fully characterize it. Another possible improvement in the paper is deployment of the STG cell on neuromorphic hardware. Input-routed graded payloads may consume more energy than binary emissions, a hardware-dependent tradeoff shared by other graded-spike models, while STG also supports binary operation.

9 Conclusion

In conclusion, we propose an approach to training SNNs that (i) expresses neural nonlinearity through TG [28] and (ii) increases gate sharpness τ to move from ANN-like behavior toward spiking emission, using exact gradients during the smooth phase to support accurate training. The key insight is that finite output resolution does not require infinite gate sharpness. We additionally propose input-routed payloads to support gradient flow and improve performance. Results on several event-driven datasets demonstrate the benefits of this approach and motivate further work on training spiking networks through smooth-to-sharp transitions.

AI use statement

In this work, we used generative AI tools to assist in writing the manuscript, such as rewording phrases and preparing LaTeX tables; to implement and run code released with previous papers; to

cross-check mathematics; and to write the appendix. We have not used generative AI tools for the core concept, its mathematical formulation, the underlying idea, or the initial draft of the paper. We have reviewed all AI-assisted work, and we take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

Reproducibility statement

The cell equations are given in Section 2 and Figure 2, with parameter constraints and numerical safeguards in Appendix A. Appendix B lists, for each benchmark, the backbone, cell configuration, sharpness schedule, quantization settings, augmentation, and evaluation protocol, including the use of test-set selection on SHD. Proofs of all formal statements are in Appendices D to H. Unless marked otherwise, results are means and standard deviations over three seeds. We will release code and trained models upon publication.

References

[1] Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo Di Nolfo, Tapan Nayak, Alexander Andreopoulos, Guillaume Garreau, Marcela Mendoza, Jeff Kusnitz, Michael Debole, Steve Esser, Tobi Delbruck, Myron Flickner, and Dharmendra Modha. A low power, fully event-based gesture recognition system. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7243–7252, 2017.

[2] Guillaume Bellec, Darjan Salaj, Anand Subramoney, Robert Legenstein, and Wolfgang Maass. Long short-term memory and learning-to-learn in networks of spiking neurons. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 795–805, 2018.

[3] Tong Bu, Wei Fang, Jianhao Ding, PengLin Dai, Zhaofei Yu, and Tiejun Huang. Optimal ANN-SNN conversion for high-accuracy and ultra-low-latency spiking neural networks. In *International Conference on Learning Representations*, 2022.

[4] Yongqiang Cao, Yang Chen, and Deepak Khosla. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 113(1):54–66, 2015. doi: 10.1007/s11263-014-0788-3.

[5] 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.

[6] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, Yuyun Liao, Chit-Kwan Lin, Andrew Lines, Ruokun Liu, Deepak Mathaikutty, Steven McCoy, Arnab Paul, Jonathan Tse, Guruguhanathan Venkataramanan, Yi-Hsin Weng, Andreas Wild, Yoonseok Yang, and Hong Wang. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018. doi: 10.1109/MM.2018.112130359.

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

[8] Albert Gu, Karan Goel, Ankit Gupta, and Christopher Ré. On the parameterization and initialization of diagonal state space models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[9] Yufei Guo, Liwen Zhang, Yuanpei Chen, Xinyi Tong, Xiaode Liu, YingLei Wang, Xuhui Huang, and Zhe Ma. Real spike: Learning real-valued spikes for spiking neural networks. In *European Conference on Computer Vision (ECCV)*, pp. 52–68, 2022. doi: 10.1007/978-3-031-19775-8_4.

[10] 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 (ICLR)*, 2024.

[11] Soufiane Hayou, Arnaud Doucet, and Judith Rousseau. On the impact of the activation function on deep neural networks training. In *International Conference on Machine Learning (ICML)*, volume 97, pp. 2672–2680, 2019.

[12] Navdeep Jaitly and Geoffrey E. Hinton. Vocal tract length perturbation (VTLP) improves speech recognition. In *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[13] Lukas König, Manuel Kuhn, David Kappel, and Anand Subramoney. Training event-based neural networks with exact gradients via differentiable ODE solving in JAX. In *Neuro Inspired Computational Elements (NICE)*, pp. 1–8, 2026. doi: 10.1109/NICE69539.2026.11567462.

[14] 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.

[15] Yuhang Li, Youngeun Kim, Hyoungseob Park, Tamar Geller, and Priyadarshini Panda. Neuromorphic data augmentation for training spiking neural networks. In *European Conference on Computer Vision (ECCV)*, 2022. doi: 10.1007/978-3-031-20071-7_37.

[16] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A ConvNet for the 2020s. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[17] Ruichen Ma, Liwei Meng, Guanchao Qiao, Ning Ning, Yang Liu, and Shaogang Hu. I2E: Real-time image-to-event conversion for high-performance spiking neural networks. *arXiv preprint arXiv:2511.08065*, 2025. URL https://arxiv.org/abs/2511.08065.

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

[19] Mark Mazumder, Sharad Chitlangia, Colby Banbury, Yiping Kang, Juan Ciro, Keith Achorn, Daniel Galvez, Mark Sabini, Peter Mattson, David Kanter, Greg Diamos, Pete Warden, Josh Meyer, and Vijay Janapa Reddi. Multilingual spoken words corpus. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2021.

[20] Paul A. Merolla, John V. Arthur, Rodrigo Alvarez-Icaza, Andrew S. Cassidy, Jun Sawada, Filipp Akopyan, Bryan L. Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, Bernard Brezzo, Ivan Vo, Steven K. Esser, Rathinakumar Appuswamy, Brian Taba, Arnon Amir, Myron D. Flickner, William P. Risk, Rajit Manohar, and Dharmendra S. Modha. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197): 668–673, 2014. doi: 10.1126/science.1254642.

[21] Ştefan Mihalaş and Ernst Niebur. A generalized linear integrate-and-fire neural model produces diverse spiking behaviors. *Neural Computation*, 21(3):704–718, 2009. doi: 10.1162/neco.2008.12-07-680.

[22] Jose Marie Antonio Miñoza. UltraLIF: Fully differentiable spiking neural networks via ultradiscretization and max-plus algebra. In *International Conference on Machine Learning (ICML)*, 2026. URL https://arxiv.org/abs/2602.11206.

[23] Emre O. Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019.

[24] Thomas Nowotny, James P. Turner, and James C. Knight. Loss shaping enhances exact gradient learning with Eventprop in spiking neural networks. *Neuromorphic Computing and Engineering*, 5(1):014001, 2025. doi: 10.1088/2634-4386/ada852.

[25] Garrick Orchard, E. Paxon Frady, Daniel Ben Dayan Rubin, Sophia Sanborn, Sumit Bam Shrestha, Friedrich T. Sommer, and Mike Davies. Efficient neuromorphic signal processing with Loihi 2. In *IEEE Workshop on Signal Processing Systems (SiPS)*, pp. 254–259, 2021. doi: 10.1109/SiPS52927.2021.00053.

[26] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in Neuroscience*, 11:682, 2017.

[27] Mark A. Rutherford, Henrique von Gersdorff, and Juan D. Goutman. Encoding sound in the cochlea: From receptor potential to afferent discharge. *The Journal of Physiology*, 599(10): 2527–2557, 2021. doi: 10.1113/JP279189.

[28] Muhammad Sabih, Frank Hannig, and Jürgen Teich. Rethinking neural nonlinearity as gating, 2026. URL `https://arxiv.org/abs/2607.03148`.

[29] Samuel S. Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep information propagation. In *ICLR*, 2017.

[30] 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.

[31] Sumit Bam Shrestha and Garrick Orchard. SLAYER: Spike layer error reassignment in time. In *Advances in Neural Information Processing Systems*, 2018.

[32] Sumit Bam Shrestha, Jonathan Timcheck, Paxon Frady, Leobardo Campos-Macias, and Mike Davies. Efficient video and audio processing with Loihi 2. In *ICASSP*, pp. 13481–13485, 2024.

[33] Jiaqi Wang, Liutao Yu, Xiongri Shen, Sihang Guo, Chenlin Zhou, Leilei Zhao, Yi Zhong, Zhiguo Zhang, and Zhengyu Ma. SpikCommander: A high-performance spiking transformer with multi-view learning for efficient speech command recognition. In *AAAI Conference on Artificial Intelligence*, 2026.

[34] Timo C. Wunderlich and Christian Pehle. Event-based backpropagation can compute exact gradients for spiking neural networks. *Scientific Reports*, 11:12829, 2021. doi: 10.1038/s41598-021-91786-z.

[35] Jason Yik, Korneel Van den Berghe, Douwe den Blanken, Younes Bouhadjar, Maxime Fabre, Paul Hueber, et al. The NeuroBench framework for benchmarking neuromorphic computing algorithms and systems. *Nature Communications*, 16:1545, 2025. doi: 10.1038/s41467-025-56739-4.

[36] Donghyung Yoo and Doo Seok Jeong. DTS-SNN: Spiking neural networks with dynamic time-surfaces. *IEEE Access*, 10:102659–102668, 2022. doi: 10.1109/ACCESS.2022.3209671.

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

[38] Tianqing Zhang, Kairong Yu, Xian Zhong, Hongwei Wang, Qi Xu, and Qiang Zhang. STAA-SNN: Spatial-temporal attention aggregator for spiking neural networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13959–13969, 2025.

[39] Chenlin Zhou, Han Zhang, Zhaokun Zhou, Liutao Yu, Liwei Huang, Xiaopeng Fan, Li Yuan, Zhengyu Ma, Huihui Zhou, and Yonghong Tian. QKFormer: Hierarchical spiking transformer using Q-K attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

A Implementation Details

Parameter constraints. The coefficient on the previous membrane in the integration branch is kept in $(\lambda_{\min}, 1)$ by $s_{h,1} = \lambda_{\min} + (1-\lambda_{\min})\sigma(\tilde{s})$, with $\lambda_{\min}$=0.01. Base thresholds for $K > 2$ use cumulative softplus increments, which keeps them ordered, and the membrane and adaptation gate sharpnesses use $1 + \mathrm{softplus}(\tilde{\tau})$. The remaining state-update coefficients are unconstrained by default. Because $s_{h,2}$, $s_{r,1}$, and $s_{r,2}$ multiply the carried state, a value above one makes that branch amplify its state at every step; an optional bound clips them to $[0, 1]$. We enable it for DVS-Gesture (T=300), where $s_{h,2}$ otherwise exceeded one and the membrane diverged while the saturated emission hid it; the bound leaves the trained model's output unchanged. Bounding these coefficients alone does not prove contraction of the full gated recurrence.

Numerical safeguards. The implementation clips the sigmoid arguments to $[-30, 30]$. Automatic differentiation then follows the implemented forward operations; the smoothness statement in Appendix F applies to the unclipped equations. An optional clamp on the emission gate's backward factor exists in the implementation but changes the derivative and is not part of the exact-gradient formulation. Because that factor is proportional to τ_o, on DVS-Gesture we scale the learning rate by $\tau_o^{\min}/\tau_o$ as the gate sharpens.

Adaptation state. With the adaptation couplings initialized at zero, ρ=0 keeps r from reaching the output and $w_{r,j}$=$c_{r,j}$=0 keeps it from being driven, and $r_0 = 0$ makes both pathways inactive. Under these conditions, task-loss gradients cannot activate either pathway and $r_t \equiv 0$. This occurred in the SSC and MSWC-FSCIL models, which therefore use only the membrane state. The SHD and CIFAR10-DVS models use a different implementation and learn nonzero couplings.

Emission-threshold timing. The choice of ℓ affects the emission threshold: $\ell = 0$ reads the updated states and $\ell = 1$ reads the previous states. Both choices leave the membrane and adaptation gates unchanged. CIFAR10-DVS uses lagged thresholds, whereas MSWC-FSCIL and the DTS-SNN DVS-Gesture model use current-state thresholds. Proposition 2 gives the corresponding algebraic relation. A controlled comparison of the two threshold timings is reported in Appendix C.

B Experimental Settings and Evaluation

SHD. DCLS-Delays uses two hidden layers, each with 256 or 384 units, and 25 learnable delay taps. The STG cell has three output branches with trainable slopes and no enforced silent branch. Over 150 training epochs, ζ is scheduled from 0.05 to 0.99 within the sharpness range $[1, 10]$. Both STG and LIF receive time masking and random resampling along the time and frequency axes. The frequency perturbation is motivated by vocal-tract-length perturbation [12]. SHD models are selected using test accuracy, without a separate validation set; Table 3 distinguishes unquantized and quantized results. Averages from epoch 100 onward are $95.65 \pm 0.46\%$ for STG at width 256 and 93.81% for LIF. The quantized result uses the width-384 model after fine-tuning with W8A8 synapses and 4-bit emission. Its event-rate deviation is measured across the same three seeds.

SSC. The backbone uses five S4D layers of width 512 and state dimension 64. The cell uses $K = 2$, $\beta_g = 1$, $\beta_p = 0$, free output slopes, and no fixed silent branch. The sharpness schedule increases ζ from 0.05 to 0.99, with midpoint epoch 30 and $\tau_o^{\max} = 50$. Quantized fine-tuning uses W8A8 synapses and 4-bit emission for 150 epochs. The primary SSC comparison uses 8-bit emission, 12 epochs of quantized fine-tuning and a dead zone of 0.078. SpikCommander's $T = 100$ result is the temporal comparison; its longer-sequence results reach 85.52% at $T = 200$ and 85.98% at $T = 250$ [33]. The paired curriculum comparison reports $84.40 \pm 0.24\%$ from the training evaluations; standalone evaluation of the same three selected models gives the $84.46 \pm 0.23\%$ unquantized reference. The curriculum contrast uses the paired training evaluations rather than mixing protocols.

CIFAR10-DVS. ConvNeXt-Small uses pretrained weights, $T = 16$, and 36 STG cells with $K = 2$ and input routing. The floating-point evaluation uses finite-sharpness sigmoid gates and unquantized affine input payloads, not hard binary or fixed multi-level emission. Training schedules ζ from 0.05 to 0.998 with $\tau_o^{\max} = 10$. Each of three independently trained models initializes its own 20-epoch quantized fine-tuning run: one epoch calibrates the quantizers and 19 epochs train with quantization active, at learning rate 2×10^{-5}. Sharpness is fixed during fine-tuning. Weights and activations use eight bits in 40 convolutional and 73 linear layers; all STG outputs use a 4-bit affine quantizer with one scale per tensor. Evaluation uses exponential-moving-average weights selected from epochs with quantization active. The train/test split is fixed across seeds. Because pretraining and the split affect comparability, the published results listed with this model in Table 3 provide context rather than a matched-capacity comparison.

DVS-Gesture. The DTS-SNN comparison retains the dynamic time-surface front end, encoder, input LIF, and readout, replacing the neurons at the hidden and output fully connected layers with STG. Both models use the same 200-epoch budget, batch size 16, $T = 300$, and 5-ms bins. The input pipeline produces 1,176 training and 288 test clips. We apply erasing, rotation, and polarity

inversion using an augmentation pipeline based on NDA [15]. Horizontal reflection is excluded because it changes the identity of directional gestures. STG has $K = 2$, fixes $s_1 = s_2 = 0$ and $c_1 = 0$, and learns c_2; the silent branch is held fixed throughout training. Its state coefficients are bounded to $[0, 1]$, and the learning rate is scaled inversely with emission sharpness. The scheduled ζ increases from 0.05 to 0.99. Over the final five epochs, STG and the reference achieve 95.07% and 91.81%, respectively; Table 3 gives their peak accuracies. This experiment uses one seed and does not include a quantized-output event-rate measurement.

MSWC-FSCIL. STG replaces the RadLIF units in the recurrent SNN of the NeuroBench few-shot class-incremental learning task. It uses $K = 2$, $\tau_o^{\max} = 10$, and a learned sharpness curriculum with $\alpha_{\max} = 1$, $\nu_\alpha = 0.15$, $f_0 = 0.5$, and sharpness learning rate 0.05. The trained output has zero slopes and one silent branch, giving binary emission in the sharp limit. The metric is mean accuracy over ten incremental sessions, not base-class accuracy. Floating-point results use three seeds; the 4-bit emission result uses one seed. Support-set sampling varies between evaluations, so the single-seed quantized value should not be subtracted from the three-seed floating-point mean as a paired quantization effect. Parameter counts are omitted because the saved model includes prototype-readout tensors as well as network parameters.

C Additional Comparisons

Curriculum dependence on the architecture. A separate DVS-Gesture experiment uses a CIFAR-style ResNet-20 with $T = 12$ and 40 training epochs. Across three seeds, mean accuracies are 80.81% with a hard gate and surrogate gradients, 84.98% with a static soft gate, 86.62% with a fixed sharp gate, and 83.84% with annealing. Fixed sharpness performs best in this setting, in contrast to the SSC comparison in Section 6.2. The DTS-SNN results in Table 3 compare complete cell and training configurations; they do not isolate the effect of annealing.

Emission-threshold timing. On ResNet-20 with DVS-Gesture, lagged thresholds improve accuracy by 1.5–3.2 points across the tested training variants and three seeds. Fixed sharpness with lagged thresholds reaches $89.77 \pm 1.36\%$. This comparison concerns the threshold-timing choice described in Appendix A, not the DTS-SNN models in Table 3.

Binary constraints and graded output. The SHD comparison between binary and graded emission is sensitive to the training configuration. Separate $K = 3$ experiments obtain $93.99 \pm 0.70\%$ with a silent branch and $92.87 \pm 1.09\%$ with fixed-level output. These use different training settings from the unquantized width-384 reference ($97.08 \pm 0.27\%$), so their accuracy gaps cannot be attributed solely to the output constraints. They motivate reporting binary and graded configurations separately, rather than assuming that enforcing binary output is cost-free on every task.

Precision and the zero interval. At a fixed explicit zero interval on SSC, 6-bit and 8-bit emission give event rates of 11.84% and 11.69%, with an accuracy difference of 0.02 points. This distinguishes the effect of the zero interval from the effect of bit width. In the primary 8-bit SSC configuration, W8A8 synapses and a zero interval set by 0.078 give $83.91 \pm 0.37\%$ at a $23.66 \pm 1.95\%$ event rate after 12 epochs of fine-tuning. The 4-bit result in Table 4 uses a longer fine-tuning budget, so these values describe two operating points, not an isolated bit-width ablation.

D Cell Equivalences

Theorem 1 (Representative cell equivalences)**.** *Let K=2 and let binary emission mean $(s_1, c_1) = (0, 0)$ and $(s_2, c_2) = (0, 1)$. (a) With β_g=1, binary emission, κ=ρ=0, and $\tau_o, \tau_h \to \infty$, the membrane settings of (11) recover* LIF *with hard, subtractive, or no reset. (b) With β_g=1, current-state emission (ℓ=0, κ=0), binary emission, and hard emission and reset gates, an emission-driven adaptation channel with $\tau_r \to \infty$ recovers spike-adaptive* LIF*; adding a recurrent input map gives its recurrent extension. (c) Finite τ_h smoothly blends the integration and reset branches. (d) With β_g=β_p=0 and κ=ρ=0, the states do not reach the output and the cell reduces to* TG *with its chosen affine emission branches: this includes* SiLU*, the sigmoid approximation to* GELU*, and* ReLU *at its sharp limit.*

Proof. At $\tau_o, \tau_h \to \infty$ the gates are $\{0,1\}$ almost everywhere. With $\beta_g{=}1$ and $\kappa{=}\rho{=}0$, $g_{h,t} = \mathbf{1}[z_t > 0]$ equals the emission of step $t{-}1$. Write h_t for the quantity compared against the threshold at step t, and let the reference be the standard reset-at-spike convention $h_t^{\text{ref}} = \lambda V_{t-1} + (1{-}\lambda)x_t$, $S_t = \mathbf{1}[h_t^{\text{ref}} > V_{\text{th}}]$, with V_t the post-reset carried state.

(a) With $\kappa{=}\rho{=}0$ the STG threshold is constant, so the two threshold timings coincide. *No reset*: identical branches give $h_t = \lambda h_{t-1} + (1{-}\lambda)x_t$ regardless of $g_{h,t}$. *Subtractive*: branch 2 of (11) gives $h_t = \lambda h_{t-1} + (1{-}\lambda)x_t - \lambda V_{\text{th}} g_{h,t} = \lambda(h_{t-1} - S_{t-1}V_{\text{th}}) + (1{-}\lambda)x_t$, which is the reference with $V_{t-1} = h_{t-1} - S_{t-1}V_{\text{th}}$. *Hard reset*: branch 2 gives $h_t = (1{-}\lambda)x_t + \lambda V_{\text{reset}}$ when $S_{t-1}{=}1$ and the continue branch otherwise, i.e. the reference with $V_{t-1} = (1{-}S_{t-1})h_{t-1} + S_{t-1}V_{\text{reset}}$. Omitting the factors of λ in (11) would instead subtract the full V_{th} after the leak, differing from the reference by $(1{-}\lambda)V_{\text{th}}$ per spike, and would drop x_t entirely on a hard-reset step.

(b) For spike-triggered adaptive LIF with adaptation decay $\gamma{=}e^{-\Delta t/\tau_a}$ and strength β_{ad}, set $\kappa{=}0$ and $\rho{=}\beta_{\text{ad}}$. With $\kappa{=}0$ and $\ell{=}0$ the state-update margin z_t equals the emission margin of step $t{-}1$, so in the hard limit $g_{r,t} = q_{t-1}^{(1)}$. Choosing $s_{r,1}{=}s_{r,2}{=}\gamma$, $w_{r,1}{=}w_{r,2}{=}0$, $c_{r,1}{=}0$, and $c_{r,2}{=}1{-}\gamma$ in (5) gives $r_t = \gamma r_{t-1} + (1{-}\gamma)q_{t-1}^{(1)}$, and the emission threshold is $\theta_{\text{STG},t} = V_{\text{th}} + \beta_{\text{ad}} r_t$. For the recurrent extension, the total input is $f(x_t){=}x_t + \mathbf{V}q_{t-1}^{(1)}$, where $\mathbf{V}$ is the recurrent weight matrix. General AdLIF and RadLIF cells may additionally couple adaptation directly to the membrane; our separate adaptation channel does not include that term.

(c) At finite τ_h, $g_{h,t} \in (0,1)$ and (4) is a convex combination of the two branches, interpolating the settings of (a).

(d) With $\beta_g{=}\beta_p{=}0$, both the gate signal and the payload equal x_t, and with $\kappa{=}\rho{=}0$ the thresholds are constant, so (7) reduces to (1) regardless of the state trajectory. With branches $(0,0)$ and $(1,0)$ and a zero threshold, $x\sigma(x)$ is SiLU, $x\sigma(1.702x)$ approximates GELU, and the sharp limit gives ReLU [28].

Membrane-reset settings.

mechanism	$s_{h,1}$	$w_{h,1}$	$c_{h,1}$	$s_{h,2}$	$w_{h,2}$	$c_{h,2}$
no reset	λ	$1{-}\lambda$	0	λ	$1{-}\lambda$	0
subtractive reset	λ	$1{-}\lambda$	0	λ	$1{-}\lambda$	$-\lambda V_{\text{th}}$
hard reset	λ	$1{-}\lambda$	0	0	$1{-}\lambda$	λV_{reset}

(11)

Substituting $g_{h,t} \in \{0,1\}$ into (4): row 1 gives $h_t = \lambda h_{t-1} + (1{-}\lambda)x_t$ independently of firing; row 2 subtracts λV_{th} on the step after a spike, equivalent to subtracting V_{th} from the membrane before the leak; row 3 replaces the leaked membrane with λV_{reset} while still integrating x_t.

D.1 Substituting the settings in Table 2

Table 2 specifies the affine state branches $a_{u,t}^{(j)} = s_{u,j}u_{t-1} + w_{u,j}x_t + c_{u,j}$. All recurrences use matched initial states, with S_0 defined by the initial membrane and threshold. The settings below specify the reset and adaptation conventions used in these equivalences.

For the ANN rows, $\beta_g = \beta_p = 0$, $\kappa = \rho = 0$, $\theta_{\text{TG}} = 0$, and $(s_1, c_1) = (0,0)$, $(s_2, c_2) = (1,0)$. Writing $q_t = q_t^{(1)}$ for the single threshold gate, Equation (7) becomes

$$y_t = (1 - q_t)0 + q_t x_t = x_t \sigma(\tau_o x_t).$$

Thus $\tau_o = 1$ gives SiLU, 1.702 gives the sigmoid approximation to GELU, and $\tau_o \to \infty$ gives $\max(0, x_t)$. The state branches are irrelevant because neither state enters this output.

For the LIF row, set $(s_{h,1}, w_{h,1}, c_{h,1}) = (\lambda, 1-\lambda, 0)$ and $(s_{h,2}, w_{h,2}, c_{h,2}) = (0, 1-\lambda, 0)$. All adaptation coefficients are zero. In the hard-gate limit, away from threshold ties, the state gate equals the previous unit spike S_{t-1}. Substitution yields

$$h_t = (1 - S_{t-1})(\lambda h_{t-1} + U_t) + S_{t-1}U_t = \lambda(1 - S_{t-1})h_{t-1} + (1-\lambda)x_t, \qquad S_t = \mathbf{1}[h_t > V_{\text{th}}].$$

This is LIF with reset to zero after the previous spike and integration of the current input. The output branches give $y_t = S_t$.

The spike-adaptive row keeps these membrane and output branches but sets $(s_{r,1}, w_{r,1}, c_{r,1}) = (\gamma, 0, 0)$ and $(s_{r,2}, w_{r,2}, c_{r,2}) = (\gamma, 0, 1-\gamma)$. With current-state thresholds, both hard state gates still equal S_{t-1}, giving

$$r_t = (1-S_{t-1})R_t + S_{t-1}(R_t + 1 - \gamma) = \gamma r_{t-1} + (1-\gamma)S_{t-1}, \qquad S_t = \mathbf{1}[h_t > V_{\text{th}} + \rho r_t].$$

Replacing x_t by $x_t + \mathbf{V}S_{t-1}$ gives the recurrent row. These are spike-triggered threshold-adaptation models; equivalence to models with additional membrane-to-adaptation coupling is not claimed.

For the last two rows, let $\theta_j := \theta_{\text{TG}}^{(j)}$ denote the constant thresholds, ordered as $\theta_1 < \cdots < \theta_{K-1}$, with $\theta_0 = -\infty$ and $\theta_K = +\infty$. If $\theta_{j-1} < h_t < \theta_j$, the hard branch weights satisfy $\pi_{j,t} = 1$ and $\pi_{k,t} = 0$ for $k \neq j$. Hence

$$y_t = c_j \quad \text{(multi-level)}, \qquad y_t = s_j\big(\beta_{\text{p}} h_t + (1-\beta_{\text{p}})x_t\big) + c_j \quad \text{(graded)}.$$

The membrane reset in these examples is triggered by crossing the first threshold, as in (4); output amplitude is selected by all $K-1$ thresholds. Multi-level output requires zero slopes and distinct offsets. For graded output, β_{p} selects the payload source without changing the membrane-driven branch selection.

D.2 State rescaling and threshold timing

State-rescaling symmetry.

Proposition 2 (State-rescaling symmetry)**.** *Fix membrane-driven firing (β_{g}=1), an endpoint payload routing $\beta_{\text{p}} \in \{0,1\}$, and the margin-driven state updates (4)–(5), with freely rescalable membrane input coefficients, offsets, and payload slopes. Under current-state emission (ℓ=0), any $\kappa < 1$ can be set to zero without changing the output sequence by taking $\tilde{h} = (1-\kappa)h$, $\tilde{w}_{h,j} = (1-\kappa)w_{h,j}$, and $\tilde{c}_{h,j} = (1-\kappa)c_{h,j}$, including the same rescaling of the initial membrane. For a membrane payload, also take $\tilde{s}_k = s_k/(1-\kappa)$; for an input payload, leave s_k unchanged. All other parameters are unchanged. For lagged emission (ℓ=1), this rescaling does not generally eliminate κ.*

Proof. For ℓ=0, the emission margin is $(1-\kappa)h_t - \rho r_t - \theta_{\text{TG}}^{(k)} = \tilde{h}_t - \rho r_t - \theta_{\text{TG}}^{(k)}$. The margin used by the state-update gates similarly becomes $\tilde{h}_{t-1} - \rho r_{t-1} - \theta_{\text{TG}}^{(1)}$, so both state-update gates are preserved. Multiplying (4) by $(1-\kappa)$ gives the recurrence in $\tilde{h}$ with the stated coefficients. The adaptation trajectory is unchanged because its gate is unchanged, and the payload rescaling preserves every affine branch. Induction gives the same output at each step. For ℓ=1, the original emission margin contains $h_t - \kappa h_{t-1}$, whereas the transformed cell with $\tilde{\kappa}$=0 contains $(1-\kappa)h_t$. Their difference is $\kappa(h_t - h_{t-1})$, which need not vanish.

Thus κ is redundant under current-state emission under the rescaling assumptions of Proposition 2. In the lagged form, $h_t - \kappa h_{t-1} = (1-\kappa)h_t + \kappa \Delta h_t$ explicitly includes a membrane-increment term. This breaks the particular symmetry above; it does not establish full parameter identifiability or an accuracy advantage for lagged thresholds.

Threshold timing and state updates. The membrane and adaptation gates always use $z_t = h_{t-1} - \theta_{\text{STG},t-1}^{(1)}$. With ℓ=1, the emission reads the same stored threshold but compares it with X_t^{gate}; with ℓ=0, the emission threshold is recomputed from the updated states. The membrane and adaptation gates are evaluated before the state updates, and the emission afterwards.

For membrane-driven firing, let $m_{t-1}^{(1)} = h_{t-1} - \theta_{\text{STG},t-1-\ell}^{(1)}$ be the previous emission margin at the first threshold. Then

$$z_t - m_{t-1}^{(1)} = \begin{cases} 0, & \ell = 0, \\ -\kappa \Delta h_{t-1} - \rho\, \Delta r_{t-1}, & \ell = 1, \end{cases} \qquad \Delta u_t := u_t - u_{t-1}. \tag{12}$$

Even when the margins agree, the gate values agree only at matched sharpness (or in their common hard limit). At a shared sharpness τ, their difference is bounded by $\frac{\tau}{4}|z_t - m_{t-1}^{(1)}|$ because the sigmoid is $1/4$-Lipschitz. This distinguishes a state update driven by the firing margin from an explicitly event-driven adaptation update.

E GRADIENT PATHS

Differentiating (1) gives

$$\frac{\partial \mathrm{TG}(x)}{\partial x} = \underbrace{\sum_k \pi_k(x) s_k}_{\text{payload path}} + \underbrace{\sum_k \pi_k'(x)(s_k x + c_k)}_{\text{gate path}}. \tag{13}$$

Theorem 3 (Gradient-path decomposition)**.** *Fix β_g=1. The Jacobian of (7) splits into the two paths of (13), and (i) if β_p=0 and $s_k \neq 0$, the payload path contains terms that bypass the membrane recurrence; (ii) if β_p=1, the payload path passes through the membrane update, and its dependence on earlier inputs follows the recurrent state Jacobians; (iii) for binary emission (s_k=0), the payload path vanishes and only the gate path remains, with its dominant contribution concentrated in a band of width $O(1/\tau_o)$ around each threshold.*

Proof. Apply the product rule to (7), separating the $\partial \pi_{k,t}$ factor from the $\partial(s_k X_t^{\mathrm{pay}} + c_k)$ factor. For (i), X_t^{pay}=x_t depends on x_t and s_k locally in t; for (ii), X_t^{pay}=h_t unrolls through (4); for (iii), s_k=0 removes the payload path in both routings. For any fixed fraction of the integral of $\tau_o \sigma'(\tau_o z)$, a band of width $O(1/\tau_o)$ around zero contains that fraction, by the substitution $u = \tau_o z$.

F SMOOTHNESS OF THE UNROLLED CELL

Theorem 4 (Differentiable continuum)**.** *Let Φ collect the cell parameters, taken in the unconstrained coordinates used for optimization before applying the transformations in Appendix A, so that $\Phi \in \mathbb{R}^D$ for D free parameters. For fixed routing coefficients and threshold timing ℓ, and any finite sequence length T, the ideal, unclipped* STG *computation over these T steps is jointly C^∞ (infinitely differentiable) in Φ and in the input sequence, and backpropagation through time returns exact gradients at every point.*

Proof. The reparametrizations (σ, softplus, and cumulative sums) are C^∞ from $\mathbb{R}^D$ onto the constrained ranges, and the sharpness floors keep every gate argument scaled by a strictly positive τ. The affine branches in (4) and (5) and the threshold (6) are polynomial in the parameters and states; each gate $\sigma(\tau(\cdot - \theta))$ is C^∞ for $\tau > 0$; the routing maps are affine. The emission (7) is a finite sum of products of C^∞ maps, and T-fold composition preserves C^∞ on the open domain $\mathbb{R}^D$.

Remark. The ideal sigmoid approaches a hard gate only in the infinite-sharpness limit. Explicit quantization can nevertheless produce discrete emission at finite sharpness (Section 4); the smoothness theorem applies before that discrete operation.

G SHARPNESS, GRADIENT VARIANCE, AND THE CURRICULUM

The curriculum has two purposes: preserve useful gradient information while the network learns its branch assignments, then sharpen those assignments toward discrete emission. The results below distinguish variation in gate derivatives from the sensitivity of an already learned output to further sharpening.

Proposition 5 (Gate-derivative variance)**.** *Let X have a fixed probability density p, continuous and bounded near θ, with $p(\theta) > 0$. Define $D_\tau(X) = \tau\sigma'(\tau(X-\theta))$ for $\tau > 0$. As $\tau \to \infty$,*

$$\mathbb{E}[D_\tau] = p(\theta) + o(1), \qquad \mathrm{Var}[D_\tau] = \frac{\tau p(\theta)}{6} + o(\tau).$$

For fixed depth L, let $D_{\tau,1}, \ldots, D_{\tau,L}$ be independent copies. The mean divided by the standard deviation of their product is asymptotically $(6p(\theta)/\tau)^{L/2}$.

Proof. Write $\varphi(u) = \sigma'(u) = \sigma(u)(1-\sigma(u))$. Using $v = \sigma(u)$ gives

$$\int_{\mathbb{R}} \varphi(u)\, du = 1, \qquad \int_{\mathbb{R}} \varphi(u)^2\, du = \int_0^1 v(1-v)\, dv = \frac{1}{6}.$$

The change of variables $u = \tau(x-\theta)$ yields

$$\mathbb{E}[D_\tau] = \int_{\mathbb{R}} \varphi(u) p(\theta + u/\tau)\, du, \qquad \frac{\mathbb{E}[D_\tau^2]}{\tau} = \int_{\mathbb{R}} \varphi(u)^2 p(\theta + u/\tau)\, du.$$

Continuity of p gives the limits $p(\theta)$ and $p(\theta)/6$. To justify them, split each integral at $|x-\theta| = \delta$: local boundedness permits dominated convergence inside this interval, while the exponentially decaying kernels and $\int p(x)dx = 1$ make the outside contribution vanish. Subtracting $\mathbb{E}[D_\tau]^2 = O(1)$ gives the variance. Under independence,

$$\mathrm{Var}\left[\prod_{j=1}^{L} D_{\tau,j}\right] = \left(\mathbb{E}[D_\tau^2]\right)^L - \left(\mathbb{E}[D_\tau]\right)^{2L} \sim \left(\frac{\tau p(\theta)}{6}\right)^L.$$

The product mean tends to $p(\theta)^L$, giving the stated ratio.

As sharpness increases, gate derivatives become larger near a threshold but smaller away from it: their peak is $\tau/4$, and their effective width is $O(1/\tau)$. Under the fixed-density assumption, this produces rare large derivatives among many small ones. Averaging B independent derivative samples divides their variance by B. The product result is an independence model, motivated by mean-field analyses of gradient propagation [29, 11]; it is not the Jacobian of the full STG network. In particular, input-routed graded emission also has the direct payload path of Appendix E.

Proposition 6 (Sharpening away from thresholds)**.** *Fix the states, thresholds, and payload at one emission step. Let $m_k = X^{\mathrm{gate}} - \theta_k$ and $a_k = s_k X^{\mathrm{pay}} + c_k$. Suppose every threshold margin satisfies $|m_k| \geq d > 0$. If y_τ is the sigmoid-gated output and y_∞ its hard-selection counterpart, then*

$$|y_\tau - y_\infty| \leq J e^{-\tau d}, \qquad J := \sum_{k=1}^{K-1} |a_{k+1} - a_k|.$$

For a single fixed margin m, its derivative with respect to the margin satisfies $0 \leq \partial_m \sigma(\tau m) \leq \tau e^{-\tau|m|}$.

Proof. For $m \neq 0$, $|\sigma(\tau m) - \mathbf{1}[m > 0]| = (1 + e^{\tau|m|})^{-1} \leq e^{-\tau|m|}$. The branch-weight definition telescopes to

$$y_\tau = a_1 + \sum_{k=1}^{K-1} \sigma(\tau m_k)(a_{k+1} - a_k).$$

Subtract the corresponding hard-gate expression and apply the triangle inequality. Finally, $\partial_m \sigma(\tau m) = \tau e^{-\tau|m|}/(1 + e^{-\tau|m|})^2 \leq \tau e^{-\tau|m|}$.

These results motivate the order of training. Smoother gates initially spread exact derivatives across a wider range of margins, allowing updates to affect branch assignments before emission becomes sharply selective. If training then establishes margins separated from the thresholds, further sharpening changes those outputs exponentially little. The late stage can therefore refine an existing solution toward hard selection rather than requiring the same branch assignments to be learned with sharp gates from initialization.

This is a conditional optimization argument. Exact derivatives describe the smooth forward computation, but do not guarantee that it reaches a good solution. The variance proposition holds for a fixed distribution; during training, the margin distribution and the loss gradients change. It therefore does not imply that loss-gradient variance must grow throughout training. Conversely, the margin bound does not establish that training will create a margin gap or that perturbations cannot accumulate through recurrent states. Near-threshold samples can remain sensitive late in training. The empirical comparison in Section 6.2 tests the curriculum's benefit for the reported configurations.

H Almost-Everywhere Convergence of the Gate

$\sigma(\tau_o(x-\theta)) \to \mathbf{1}[x > \theta]$ almost everywhere: for $x \neq \theta$ the argument tends to $\pm\infty$ and σ to $\{1, 0\}$; the point $\{\theta\}$ has measure zero.