SPIKCTU: A COMPACT SPIKING NETWORK WITH ADAPTIVE RECURRENT COMPUTATION FOR SPEECH COMMAND RECOGNITION

Anonymous authors
Paper under double-blind review

ABSTRACT

Spiking Neural Networks offer event-driven, energy-efficient computation for speech command recognition, yet most existing methods follow fixed computation paths, allocating redundant computation to easy samples while providing limited insight into how evidence evolves during inference. To address these challenges, we introduce **SpikCTU**, a compact spiking network with adaptive recurrent computation for speech command recognition. Specifically, a hybrid spiking embedding module preserves continuous input information alongside spike features to construct a fixed temporal memory. A shared Continuous Thinking Unit (CTU) repeatedly reads this memory using keys derived from spikes and continuous values, and updates a binary recurrent state with lightweight stage adapters. Furthermore, combining a fixed base prediction with accumulated corrections at each step enables learned aggregation over all steps or confidence-based early exit using only the predictions available so far. Experiments on three speech command recognition benchmarks show that SpikCTU achieves competitive accuracy with only 0.16–0.17 M parameters, and the early exit strategy can substantially further reduce computational cost without compromising accuracy. In addition, trajectory and attention analyses reveal that recurrent steps enable error correction and progressively refine predictions. Code is available at `https://anonymous.4open.science/r/spikctu-5022/`.

1 INTRODUCTION

Spiking Neural Networks (SNNs), widely regarded as the third generation of neural networks (Maass, 1997), communicate through biologically plausible, sparse spike signals. Spike-driven synaptic operations can replace multiply–accumulate (MAC) operations in artificial neural networks (ANNs) with accumulations (ACs), potentially reducing energy consumption (Nunes et al., 2022; Hu et al., 2024; Sabbella et al., 2025). This energy advantage is further amplified on neuromorphic hardware, where sparse, event-driven spikes map directly onto low-energy asynchronous circuits (Bos & Muir, 2024; Wang et al., 2025). In recent years, spiking models have achieved competitive performance across vision and event-based benchmarks and have begun to match ANN accuracy on speech command recognition (SCR) (Wang et al., 2026a; Li et al., 2026; Wang et al., 2026b). However, the realized efficiency depends on how inference is configured: fixed timesteps, forward passes that evaluate every step, and dense spike propagation can waste computation on easy inputs, motivating conditional computation that allocates compute according to input difficulty.

Adaptive computation enables computational effort to scale with sample difficulty rather than a fixed worst-case budget. Adaptive Computation Time (ACT) (Graves, 2016), PonderNet (Banino et al., 2021), and Continuous Thought Machines (CTM) (Darlow et al., 2025) make iteration count a computational dimension, and SNNs with early exit suppress redundant timesteps by temporal confidence (Li et al., 2023b). However, adaptive recurrence is not domain-neutral in SNNs. Binarizing membrane potentials into spikes limits the information carried by each neuron (Bal & Sengupta, 2025), whereas a continuous path abandons the sparse, event-driven communication behind SNN efficiency (Nunes et al., 2022). Recurrent spike state dynamics can also cause predictions to fluctuate across steps, such that confidence-based stopping may terminate prematurely on an incorrect prediction (Li et al., 2023a; Chen et al., 2025), while relying solely on the final state can obscure both late corrections and late degradation. Thus, the key challenge is not simply to add recurrence to an

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

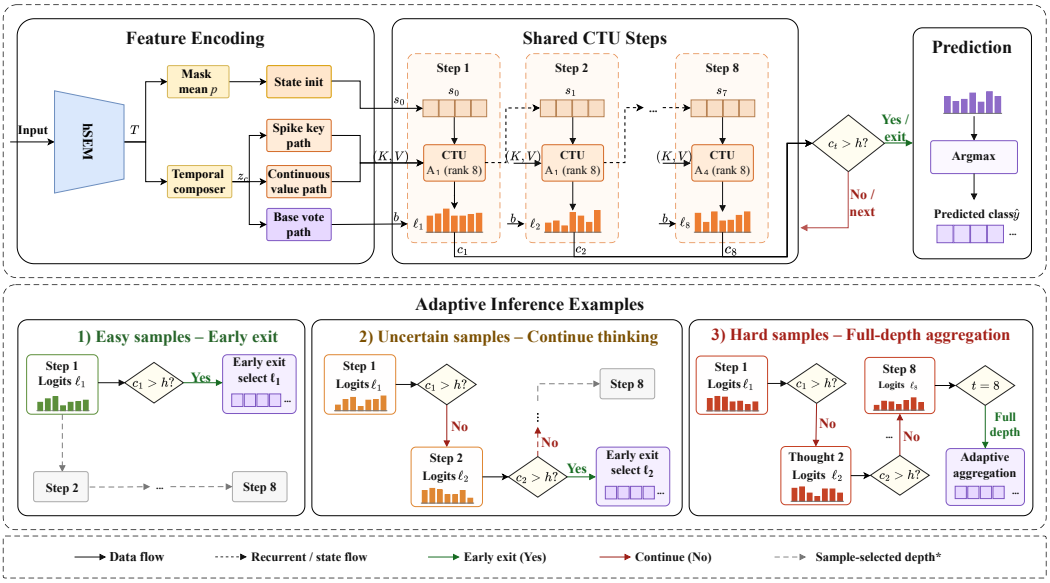

Figure 1: SpikCTU separates a fixed memory that preserves information from shared recurrent computation over a binary spike state. Each step emits evidence tied to the current prediction; inference either exits using current evidence or aggregates all steps. Grey paths denote logically skipped steps.

SNN, but to organize continuous information, spike-domain computation, and stopping decisions so that each step remains useful and auditable.

Inspired by the internal computation in CTM, we propose SpikCTU, a compact spiking architecture guided by three design principles. *Information preservation* retains continuous values in a fixed memory for repeated access. *Recurrent computation over spikes* uses a shared Continuous Thinking Unit (CTU) to update a binary spike state while maintaining membrane dynamics across steps. *Adaptive stopping from intermediate predictions* allows computation to end using predictions available at the current step. Together, these principles separate input encoding from recurrent computation and allow the number of recurrent steps to vary across samples. The resulting models combine competitive accuracy with few parameters and low estimated inference energy, addressing the memory and energy constraints of speech command recognition on edge devices.

Concretely, as illustrated in Fig. 1, SpikCTU combines a hybrid spiking embedding module (hSEM), a temporal memory with keys obtained from spikes and continuous values, and the shared CTU with low-rank stage adapters. Each recurrent step consists of one memory read, one state update, and one intermediate prediction from the shared CTU. Aggregating over all steps learns a lightweight mixture over the resulting trajectory, whereas early exit returns the first sufficiently certain prediction based on the fixed base evidence without accessing future steps. On the Spiking Heidelberg Digits, Spiking Speech Commands (Cramer et al., 2022), and Google Speech Commands V2 (Warden, 2018) benchmarks, our method achieves competitive accuracy with 0.16–0.17 M parameters. Our contributions are as follows:

- We propose a compact spiking network for energy-efficient speech command recognition. It retains continuous values in a fixed memory and reuses a shared CTU to update a binary spike state, allowing additional recurrent computation with little parameter growth.
- We design an output mechanism that refines a fixed base prediction at each recurrent step. These intermediate predictions support aggregation at full depth and early exit based on current confidence.
- Experiments on SHD, SSC, and GSC show competitive accuracy with few parameters. Further analysis examines how predictions change across recurrent steps and shows that early exit improves accuracy over a baseline with the same average number of steps on SSC and GSC.

2 Related Work

Spiking neural networks. SNNs transmit information through sparse, binary spike events rather than continuous values, and their event-driven dynamics promise low-power inference on neuromorphic hardware (Nunes et al., 2022). Modern SNNs are typically trained directly by surrogate-gradient backpropagation through stateful neuron dynamics such as leaky integrate-and-fire (LIF) neurons (Dampfhoffer et al., 2024), with learnable membrane constants adapting neuronal timescales (Fang et al., 2021). Recent SNNs have achieved competitive results in vision (Zhou et al., 2023; Yao et al., 2023) and low-power neuromorphic deployment (Bos & Muir, 2024; Wang et al., 2025). For instance, Spikformer V2 attains accuracy competitive with ANN Transformers on ImageNet-1K (Zhou et al., 2024). Nevertheless, the realized efficiency of such models depends on how information is represented, since binarizing continuous input values into spikes discards information, whereas a fully continuous path abandons the sparse communication behind SNN efficiency (Hu et al., 2024; Sabbella et al., 2025). This motivates our design, which maintains continuous input values in a fixed memory and recurrent computation over a binary spike state.

Spiking speech command recognition. Spiking SCR models process speech inputs using learned filterbanks (Song et al., 2024b) or convert them into events through temporal, pulse-density, and time-difference encoding (Nilsson et al., 2023; Yarga & Wood, 2024; Pequeño-Zurro et al., 2025). Downstream temporal structure is modeled through sparse recurrence (Dampfhoffer et al., 2023), temporal delay connections (Malettira et al., 2025; Bouhadjar & Neftci, 2026), structured-state and state-space neurons (Bal & Sengupta, 2025; Ahmed et al., 2026). Concretely, SpikeSCR (Wang et al., 2026a) obtains compact models through curriculum-based distillation, while SpikCommander (Wang et al., 2026b) achieves strong SCR accuracy with a spiking embedding module and multi-view temporal attention. Efficiency has been pursued through early decisions, as in ED-sKWS (Song et al., 2024a), activity regularization (Louis et al., 2024), compact delayed convolutions (Li et al., 2026), and asynchronous neuromorphic processors (Zhang et al., 2024). Our focus is on adapting recurrent depth after sensory encoding, while retaining continuous memory values for repeated retrieval.

Conditional computation and adaptive inference. Conditional computation allocates compute to inputs that need it (Bengio et al., 2013; Shazeer et al., 2017). In ANNs, ACT (Graves, 2016), PonderNet (Banino et al., 2021), Universal Transformers (Dehghani et al., 2019), and CTM (Darlow et al., 2025) establish recurrent depth as a controllable computation axis, but they do not address the placement of continuous information and binary state in an SNN. In SNNs, confidence-based policies shorten the timesteps of easy samples (Li et al., 2023b;a;c), reinforcement learning guides the early exits (Patne et al., 2026), and gating combines the outputs of spiking experts (Yang et al., 2026; Shim et al., 2016). Heuristic confidence thresholds can stop too early. SpikeCP instead uses conformal prediction to provide reliability guarantees at adaptive stopping times (Chen et al., 2025). These approaches adapt simulation time, network depth, or expert selection. Our focus is on adapting recurrent computational depth after sensory encoding, using a shared unit to refine predictions without extending the input sequence.

3 Method

As shown in Fig. 1, SpikCTU first builds a fixed memory from the padded event sequence X, then repeatedly reads it with a shared recurrent unit. First, the hSEM encodes X into hybrid tokens T, which are converted into the initial recurrent state s_0 and the fixed memory C. Then, C yields spike keys K, continuous values V, and fixed base evidence b. One shared CTU projects the current state to a query and reads K and V repeatedly. Each cycle of this unit, comprising one memory read, one update of the recurrent state, and one emission of refined evidence, is a recurrent step. The number of steps D defines recurrent depth. Every step updates a binary recurrent state s_t and produces a prediction based on the fixed base evidence. Stopping is causal across steps, although the fixed memory encodes the complete input.

3.1 Fixed memory that preserves information

Binarizing the input features into spikes discards their continuous information, and a continuous recurrent path loses efficiency from sparse spikes. The hSEM therefore maps the input to token width d_T by combining two LIF spike paths with a gated continuous residual. Specifically, X is embedded

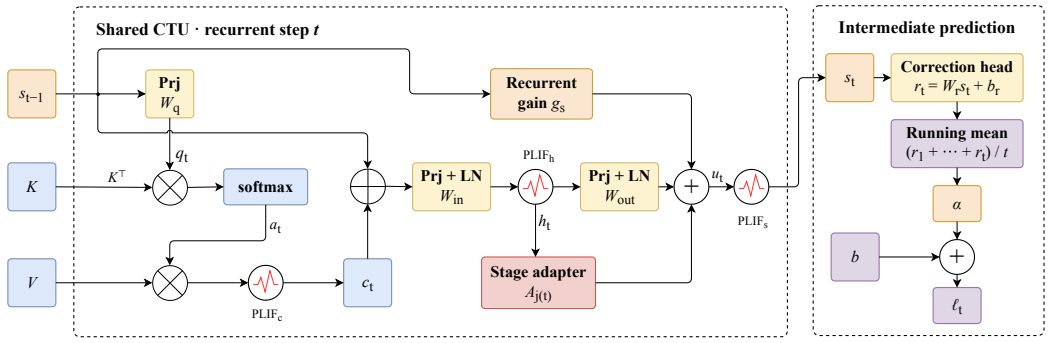

Figure 2: One recurrent CTU step. A query from the binary state s_{t-1} reads fixed spike keys K and continuous values V with multi-head attention. The shared refiner updates s_t with recurrent gain and a stage adapter. A shared classifier adds the running correction mean to base evidence b, producing ℓ_t and a causal confidence signal.

into a hidden continuous representation by pointwise and depthwise convolutions followed by batch normalization. The masked version of this representation gives the continuous residual A, while the same representation is converted into $S^{(1)}$ by a LIF, and $S^{(1)}$ then passes through a linear layer, batch normalization, and another LIF to yield $S^{(2)}$. The three branches are fused into the hybrid token:

$$T = M \odot \big(S^{(1)} + S^{(2)} + \omega(a)A\big), \tag{1}$$

where $\omega(\cdot)$ is the sigmoid function and a is learnable, with the gate $\omega(a)$ initialized to 0.05 to scale only the continuous residual. The masked mean of T yields the pooled token $p = \frac{\sum_l T_l}{\sum_l M_l}$ that initializes the recurrent state, where l indexes token positions and M_l masks padded positions.

Then T is mapped to state width d_s via a linear layer W followed by LayerNorm (LN). Temporal composition then adds local and long-range context through depthwise branches with kernel sizes $\mathcal{K} = \{7, 31\}$ and learnable gating scales ε_k. Hence, the fixed memory C is given by:

$$C = \mathrm{LN}(WT) + \sum_{k \in \mathcal{K}} \omega(\varepsilon_k)\, \mathrm{LN}_k(\mathrm{DWConv}_k(\mathrm{LN}(WT))), \tag{2}$$

The shared CTU reads this memory selectively. A token PLIF turns C into keys obtained from spikes K, while the continuous C supplies values V, so retrieval stays spike-based without binarizing the content. A head that averages token predictions on C then produces the base evidence $b = \log \frac{\sum_l M_l \,\mathrm{softmax}(W_b C_l)}{\sum_l M_l}$, anchoring later predictions to a reference independent of the recurrent state.

3.2 Shared Recurrent Computation over a Binary Spike State

CTM uses neuron-level history models and synchronization representations (Darlow et al., 2025). Our shared CTU instead reads the fixed memory using a binary state and updates it through PLIF dynamics (Fig. 2). The pooled token is mapped to state width d_s and initializes $s_0 \in \{0, 1\}^{B \times d_s}$ via the shared state PLIF (Appendix A). At step t, s_{t-1} is projected to a query $q_t = W_q s_{t-1}$ and reads the fixed K and V with multi-head attention, masking padded positions. The resulting context passes through a PLIF to obtain a spike-based c_t, which is concatenated with s_{t-1} and fed into a shared refiner of width $2d_s$. The refiner first produces $h_t = \mathrm{PLIF}_h(\mathrm{LN}(W_{\mathrm{in}}[s_{t-1}; c_t]))$ and then combines its output with a gated state connection and a stage adapter, giving

$$u_t = \mathrm{LN}(W_{\mathrm{out}} h_t) + g_s s_{t-1} + A_{j(t)}(h_t), \tag{3}$$

followed by $s_t = \mathrm{PLIF}_s(u_t)$. Here g_s is softplus-parameterized to stay positive, and $W_{\mathrm{in}}, W_{\mathrm{out}}$ are learnable refiner projections. To give later stages limited dedicated capacity without unsharing the CTU, the rank-r adapter $A_{j(t)}$ changes every two steps (one stage), while the reader, synapses, classifier, gains, and PLIF nodes remain shared. Increasing D thus adds recurrent evaluations and only low-rank adapter capacity. The resulting state s_t is passed to the shared classifier, which emits a correction to the fixed base evidence and supplies the confidence signal used for causal stopping, as shown in Fig. 2. Fig. 3(a) confirms that the binary activity pattern of s_t changes across steps rather than replaying a fixed state.

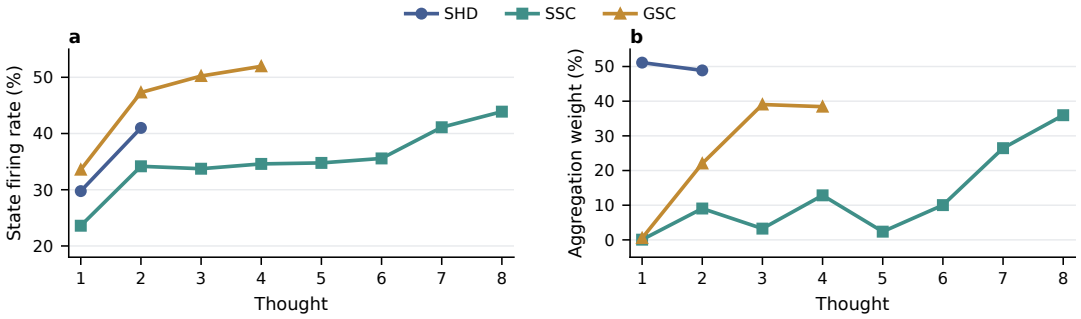

Figure 3: Diagnostics of the two learned mechanisms in the CTU. (a) Binary state activity varies across steps, indicating that the recurrent state is updated rather than replayed. (b) Aggregation assigns weights that vary across steps and tasks.

3.3 Intermediate Predictions and Adaptive Output

Persistent membrane dynamics can correct an early belief but can also damage it, so a last-state classifier hides both outcomes. Each step starts from the same base evidence and adds a running correction, giving $\ell_t = b + \alpha \frac{1}{t} \sum_{i=1}^{t} r_i$, where $r_t = W_r s_t + b_r$ and α is learnable and initialized small. The logits ℓ_t provide a causal confidence signal through normalized negative entropy, $q_t^{\text{cert}} = 1 - H(\text{softmax}(\ell_t))/\log N_c$, where H is the Shannon entropy.

At full depth, scores $e_t = w^\top s_t + \rho_t + \gamma q_t^{\text{cert}}$ combine learned state weights w, position biases ρ_t, and certainty weight γ, with $\pi_t = \text{softmax}(e_1, \ldots, e_D)_t$. A learned β, initialized small, scales the correction to the last prediction:

$$\ell_{\text{full}} = \ell_D + \beta(m - \ell_D), \qquad m = \sum_{t=1}^{D} \pi_t \ell_t. \tag{4}$$

Early exit returns the first ℓ_t satisfying $q_t^{\text{cert}} > h$ (patience one), or ℓ_D otherwise. It always bypasses aggregation, including at depth D, and uses no future steps. Fig. 3(b) shows that aggregation weights vary across steps and tasks.

3.4 Training Objective

Cross-Entropy (CE) on the output after all steps alone supervises only the final output, $\mathcal{L}_{\text{full}} = \text{CE}(\ell_{\text{full}}, y)$. To make every step useful, we add anytime supervision $\mathcal{L}_{\text{any}} = \sum_{t=1}^{D} \omega_t \text{CE}(\ell_t, y)$ whose weights assign greater total weight to later steps. Let $\varepsilon_{n,t} = \text{CE}(\ell_{n,t}, y_n)$, and let $\mathcal{C}_t$ and $\mathcal{W}_t$ be the samples correct and incorrect at step t. To preserve initially correct predictions, $\mathcal{L}_{\text{pre}}$ penalizes increases in their CE relative to step 1. For later refinement, $\mathcal{L}_{\text{ref}}$ minimizes CE on previously incorrect samples and penalizes CE increases on previously correct ones. Here avg denotes the mean over each indicated subset:

$$\mathcal{L}_{\text{pre}} = \text{avg}_{n \in \mathcal{C}_1,\, t>1} [\varepsilon_{n,t} - \varepsilon_{n,1}]_+, \tag{5}$$

$$\mathcal{L}_{\text{ref}} = \frac{1}{D-1} \sum_{t=2}^{D} \left(\text{avg}_{n \in \mathcal{W}_{t-1}} \varepsilon_{n,t} + \text{avg}_{n \in \mathcal{C}_{t-1}} [\varepsilon_{n,t} - \varepsilon_{n,t-1}]_+ \right). \tag{6}$$

Both losses are zero for $D = 1$, and empty subsets contribute zero. The total loss is $\mathcal{L} = \mathcal{L}_{\text{full}} + \lambda_{\text{any}} \mathcal{L}_{\text{any}} + \lambda_{\text{pre}} \mathcal{L}_{\text{pre}} + \lambda_{\text{ref}} \mathcal{L}_{\text{ref}}$, with the coefficients and task-specific switches given in Appendix A.

4 Experimental Setups

Datasets, protocols, and optimization. We evaluate SpikCTU on the 20-class Spiking Heidelberg Digits (SHD), the 35-class Spiking Speech Commands (SSC), and the 35-class Google Speech Commands V2 (GSC) (Cramer et al., 2022; Warden, 2018). SHD provides 8,156 training and 2,264 test utterances without an official validation split, so we follow the standard SHD protocol. SSC uses 75,466/9,981/20,382 samples, and GSC uses the official 84,843/9,981/11,005 train/validation/test splits. For SSC and GSC, we select the best performing model on validation sets and evaluate it

Table 1: Accuracy, trainable parameters, and theoretical 45-nm proxy energy. [A] denotes ANN. Bold and underlined values rank first and second within each dataset. "–" denotes unavailable results. Baseline sources are given in Appendix A.4, and multi-seed results for SpikCTU in Appendix A.3.

Data	Model	Params (M)	Acc. (%)	Energy (mJ)
SHD	DCLS-Delays (2L) (Hammouamri et al., 2024)	0.20	95.07	–
	d-cAdLIF (Deckers et al., 2024)	**0.08**	94.85	–
	SpikeSCR (1L) (Wang et al., 2026a)	0.26	95.60	<u>0.0051</u>
	Pfa-SNN (Sun et al., 2025)	0.20	96.26	–
	EventProp+Delays (Mészáros et al., 2025)	–	93.20	–
	SpikCommander (1L) (Wang et al., 2026b)	0.19	<u>96.41</u>	**0.0039**
	SpikCTU ($D = 2$)	<u>0.16</u>	**96.60**	0.0140
SSC	DCLS-Delays (3L) (Hammouamri et al., 2024)	2.50	80.69	–
	d-cAdLIF (Deckers et al., 2024)	0.70	80.23	–
	SpikeSCR (2L) (Wang et al., 2026a)	3.30	<u>82.79</u>	0.0314
	DelRec (Queant et al., 2025)	0.37	82.60	–
	SiLIF (Fabre et al., 2025)	<u>0.35</u>	82.03	–
	Pfa-SNN (Sun et al., 2025)	0.71	80.18	–
	EventProp+Delays (Mészáros et al., 2025)	–	76.10	–
	SpikCommander (2L) (Wang et al., 2026b)	2.13	**83.49**	<u>0.0180</u>
	SpikCTU ($D = 8$)	**0.17**	81.74	**0.0158**
GSC	KW-MLP[A] (Morshed & Ahsan, 2021)	<u>0.42</u>	**97.56**	0.2070
	DCLS-Delays (3L) (Hammouamri et al., 2024)	2.50	95.35	–
	d-cAdLIF (Deckers et al., 2024)	0.61	95.69	–
	SpikeSCR (1L) (Wang et al., 2026a)	1.63	95.56	0.0670
	Spiking LMUFormer (Liu et al., 2024)	1.69	96.12	0.0590
	SiLIF+Delays (Fabre et al., 2025)	0.61	95.88	–
	SpikCommander (2L) (Wang et al., 2026b)	2.13	<u>96.92</u>	<u>0.0420</u>
	SpikCTU ($D = 4$)	**0.17**	95.96	**0.0231**

on test sets; SHD follows its standard train/test protocol. For fair comparison, we set the temporal resolution $\mathsf{T} = 100$. All SpikCTU models use token width $d_T = 128$, state width $d_s = 96$, hidden ratio 2, $H = 8$ reader heads, rank $r = 8$ stage adapters, hSEM dropout 0.15, and AdamW with a cosine schedule. More implementation details and hyperparameters can be found in Appendix A.

Baselines. We compare with DCLS-Delays (Hammouamri et al., 2024), d-cAdLIF (Deckers et al., 2024), SpikeSCR (Wang et al., 2026a), Pfa-SNN (Sun et al., 2025), SiLIF (Fabre et al., 2025), DelRec (Queant et al., 2025), EventProp (Mészáros et al., 2025), Spiking LMUFormer (Liu et al., 2024), SpikCommander (Wang et al., 2026b), and the ANN baseline KW-MLP (Morshed & Ahsan, 2021). DCLS-Delays and d-cAdLIF results come directly from their original papers. Baseline energy estimates follow SpikCommander's reported comparisons. We retain published configurations rather than retraining baselines with identical preprocessing. Source details are given in Appendix A.4.

Metrics. We report top-1 accuracy and trainable parameters for predictive performance and compactness. For computational cost, we report sparse synaptic operations (SOPs) for spiking layers and multiply–accumulate operations (MACs) for continuous paths, together with a 45-nm proxy energy $E = 0.9\,\mathrm{pJ} \times \mathrm{AC} + 4.6\,\mathrm{pJ} \times \mathrm{MAC}$, following prior operation-based estimates.

5 Results

5.1 Accuracy, model size, and operation estimates

SpikCTU has the smallest reported parameter count on SSC and GSC and the highest reported accuracy on SHD among the compared methods (Table 1). On SHD, d-cAdLIF is smaller, while our 96.60% accuracy exceeds SpikCommander by 0.19 percentage points under a seed-matched protocol. On SSC, it trails the best-performing model, SpikCommander, by 1.75 points while using approximately one twelfth of its parameters. On GSC, it is 0.96 points below SpikCommander, the most accurate spiking baseline, while using only 55% of its reported energy. Among methods with energy estimates, SpikCTU has the lowest reported values on SSC and GSC. These comparisons support a compact design with competitive accuracy. Energy rankings differ on SHD, where some baselines count all synaptic operations as accumulations, while we count continuous paths as MACs. The accounting conventions and operation breakdown are given in Appendix D.

Table 2: Cross-task component ablation. Each row removes or replaces one component of the full model on the first row. Bold and underline mark the best and second-best accuracy per task. †The full SHD model trains without L_{ref}, so this row enables it on SHD.

Configuration	SHD (D=2)		SSC (D=8)		GSC (D=4)	
	Acc. (%)	Params (K)	Acc. (%)	Params (K)	Acc. (%)	Params (K)
Full	**96.60**	164.23	**81.74**	174.04	**95.96**	169.43
w/o temporal composition	94.30	160.19	78.99	170.01	94.38	165.39
w/ continuous K/V memory	95.14	164.23	81.28	174.04	95.85	169.43
w/ spike K/V memory	95.49	164.23	81.12	174.04	95.75	169.43
w/o stage adapters	94.83	161.92	81.24	164.82	95.85	164.82
w/o adaptive aggregation	94.43	164.13	81.21	173.94	95.85	169.33
w/o L_{any}	<u>95.85</u>	164.23	81.12	174.04	<u>95.90</u>	169.43
w/o L_{pre}	95.63	164.23	<u>81.49</u>	174.04	95.57	169.43
w/o L_{ref}	96.16†	164.23	81.63	174.04	95.71	169.43

5.2 Ablation studies

Temporal composition makes the largest consistent contribution across tasks. Removing it reduces accuracy by 1.58–2.75 percentage points while saving only about 4 K parameters (Table 2). Replacing the hybrid memory with either continuous keys and values or spiking keys and values also lowers accuracy at unchanged model size. These results support combining temporal context with continuous memory values and keys obtained from spikes. Stage adapters and adaptive aggregation provide further gains with limited parameter overhead. The adapters allow some specialization across recurrent stages while retaining the shared CTU. Aggregation improves on using only the last prediction, gaining 2.17 percentage points on SHD with about 0.1 K additional parameters. Thus, even a short recurrent sequence can benefit from combining intermediate predictions. Anytime supervision and preservation of the first prediction improve accuracy across all three tasks without adding parameters. Refinement loss has a less uniform effect. It helps on SSC and GSC but lowers accuracy on SHD, supporting the task-specific choice used in the full models.

5.3 Recurrent updates can correct or degrade predictions

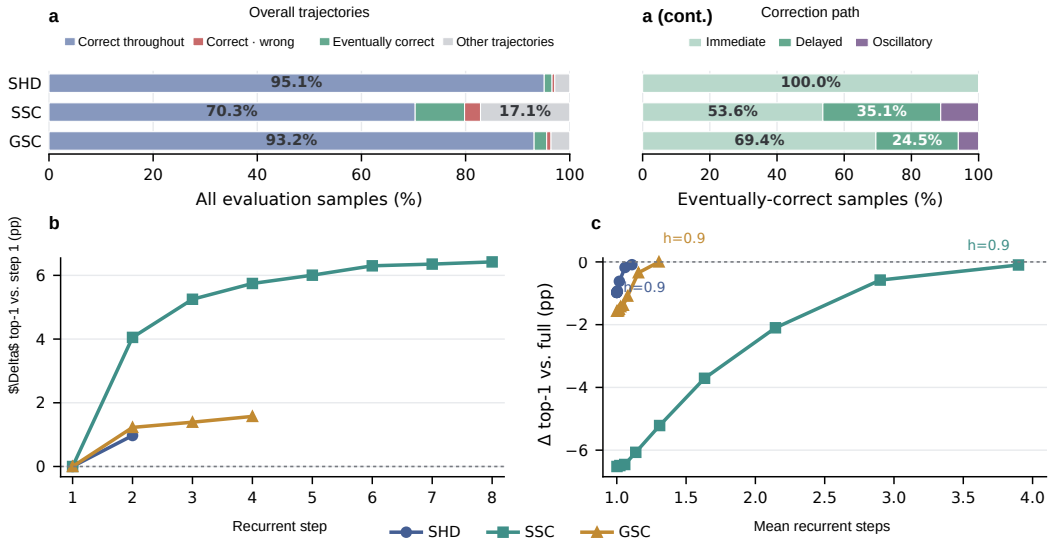

Figure 4: Prediction changes during recurrent computation. (a) Trajectory composition and correction types. Detailed counts are given in Appendix C.1. (b) Top-1 change across steps of the same model. (c) Top-1 gap to full inference under h=0.9 early exit.

Average accuracy improves across recurrent steps, but individual predictions can change in either direction (Fig. 4). On SSC, 9.48% of samples change from an incorrect initial prediction to a correct final prediction, while 3.06% change in the opposite direction. Some corrections occur only after several steps or after repeated changes in correctness. These trajectories show why the last prediction

alone provides an incomplete view of recurrent computation. Panel (b) follows successive outputs of one trained model. It should be distinguished from the depth sweep, where a separate model is trained at each depth. Detailed trajectory counts are provided in Appendix C.1. The state and aggregation diagnostics in Fig. 3 show that recurrent steps update the binary state and receive different aggregation weights. Figure 5 illustrates attention across recurrent steps for three SSC examples whose predictions remain correct, change from wrong to correct, or change from correct to wrong. Attention also moves in both corrected and degraded predictions, so we examine temporal occlusion alongside attention weights (Jain & Wallace, 2019; Wiegreffe & Pinter, 2019; Serrano & Smith, 2019). On SSC, masking the top 10% of attended bins at the final step reduces accuracy by 17.65 percentage points, compared with 4.55 points for random masking of the same proportion. Together, these results indicate that the model uses the attended input regions, while changes in attention alone do not establish that a prediction has improved (Appendix C.2).

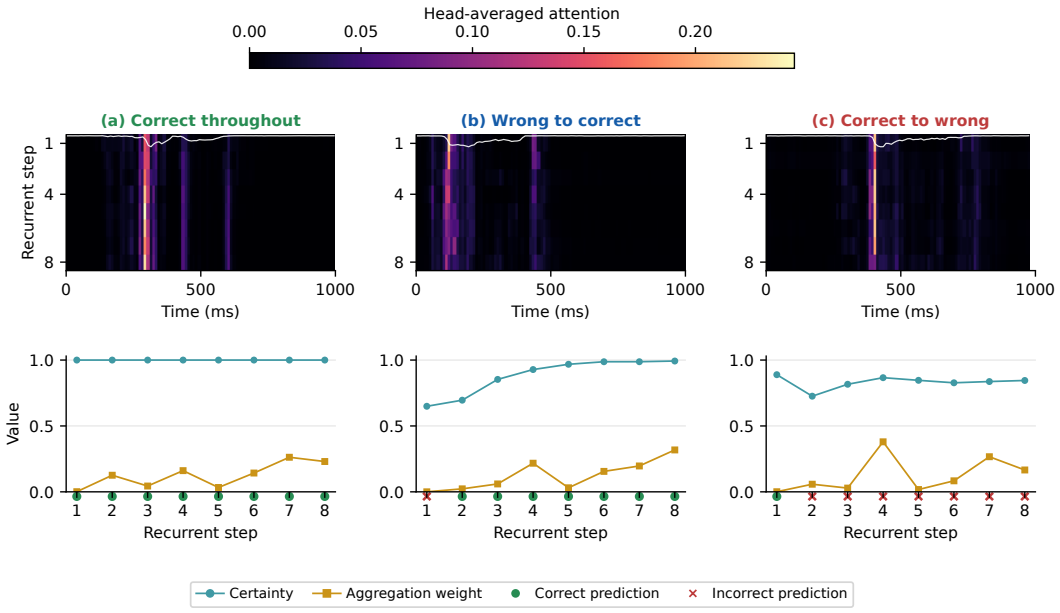

Figure 5: Sample-level attention across recurrent steps on SSC. Columns show predictions that (a) remain correct, (b) change from wrong to correct, and (c) change from correct to wrong. Top: attention maps. Bottom: certainty, aggregation weight, and correctness.

5.4 Adaptive computation with selected thresholds

Early exit reduces recurrent computation by 59% on SSC and 67% on GSC (Table 3). At the same average number of steps, it improves accuracy over the interpolated baseline by 0.81 and 1.21 percentage points, respectively. SHD shows no gain over this baseline at its shallow selected depth. The comparison suggests that intermediate confidence can allocate recurrent computation more effectively than uniform truncation on SSC and GSC. Thresholds are selected on calibration data with an allowed accuracy loss of at most 0.1 percentage points. The baseline interpolates between adjacent truncation depths of the same frozen model, and exited predictions use no future steps. Initial prediction uncertainty is also informative about exit depth, although a coarse allocation based only on this uncertainty performs less well than confidence stopping (Appendix B.3). The reported savings concern recurrent evaluations. Encoding and memory construction remain fixed and dominate estimated energy, so the reduction in total inference energy is smaller (Appendix D).

Table 3: Adaptive computation with thresholds selected on validation data: accuracy, mean evaluated steps, and gain over a fixed-depth result at the same average compute.

Data	D	h	Acc. (%)	Mean steps	Reduction	Gain (pp)
SHD	2	0.01	95.63	1.00	50.00%	+0.00
SSC	8	0.84	81.42	3.28	59.04%	+0.81
GSC	4	0.91	95.99	1.32	66.97%	+1.21

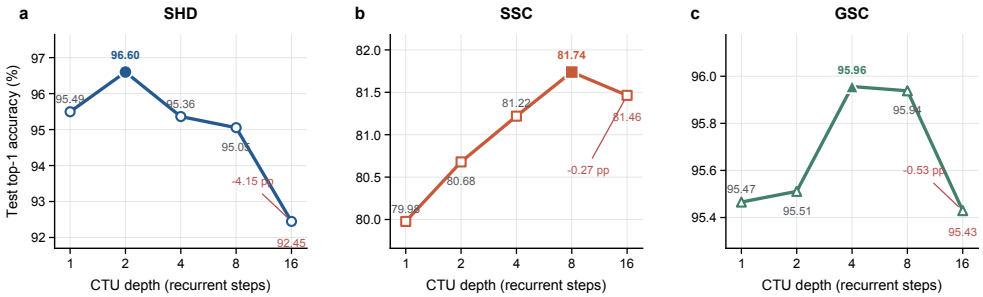

Figure 6: Top-1 accuracy versus recurrent CTU depth on SHD, SSC, and GSC. Each point is trained and evaluated at the indicated number of steps. The different optima show that additional recurrent computation does not always improve accuracy.

5.5 Depth and resolution depend on the task

The best recurrent depths are $D = 2, 8, 4$ for SHD, SSC, and GSC, respectively (Fig. 6). Further increases reduce accuracy, most clearly on SHD, where $D = 16$ falls 4.15 percentage points below the optimum. Because the CTU is shared, increasing depth adds repeated evaluations with only modest parameter growth. More recurrent computation is therefore useful only up to a task-dependent depth, and depth should be selected rather than increased by default. Sensory resolution has a different role. Table 4 shows that increasing T from 50 to 100 improves all three tasks, whereas increasing it to 250 helps only SSC. While T controls the granularity and cost of input encoding, D controls the number of recurrent updates after encoding. We use $\mathsf{T} = 100$ for the main accuracy and energy comparisons and report the full sweep in Appendix B.2. Each configuration uses one controlled run, so these sweeps describe sensitivity rather than statistical uncertainty.

Table 4: Top-1 accuracy (%) at different sensory resolutions. $\Delta = \mathrm{Acc}_{250} - \mathrm{Acc}_{100}$, in percentage points.

Data	$\mathsf{T} = 50$	$\mathsf{T} = 100$	$\mathsf{T} = 250$	Δ
SHD	95.01	**96.60**	94.66	-1.94
SSC	79.78	81.74	**82.53**	$+0.79$
GSC	95.25	**95.96**	95.44	-0.52

6 Limitations

Our study has three methodological limitations. First, the CTU reads a fixed, composed memory using softmax attention over keys obtained from spikes and continuous values. Future work could therefore investigate writable sparse memory and matrix-free reading, since adaptive depth does not remove the fixed cost of constructing and retaining the composed memory. Second, the halting rule is calibrated after training rather than learned under an explicit compute budget, and the hyperparameters of the recurrent loop are fixed rather than systematically swept. Post-hoc calibration may not transfer across sensory resolutions or deployment constraints. Third, our evidence is limited to supervised closed-set speech classification. Evaluation on streaming settings, open-vocabulary tasks, and other event modalities is needed to clarify the limits of transferability.

7 Conclusion

SpikCTU presents a compact way to organize information and recurrent computation in an SNN. Continuous values remain available in a fixed memory, while binary states support repeated updates and intermediate predictions support adaptive stopping. Across SHD, SSC, and GSC, the resulting models use 0.16–0.17 M parameters and achieve a favorable balance between accuracy and model size. Our analyses of prediction trajectories make the recurrent process visible, showing that later steps can correct an initial prediction, reinforce it, or occasionally degrade it. Calibrated stopping further shows that recurrent depth can be allocated to individual samples, improving the accuracy–computation trade-off on SSC and GSC. Sensitivity studies distinguish the roles of sensory resolution and recurrent depth, with T controlling the granularity of encoded evidence and D controlling computation after encoding. Together, these results establish recurrent depth as a useful axis for adaptive spike-domain sequence computation and point toward policy-consistent training, writable memory, and hardware evaluation.

REFERENCES

Tauseef Ahmed, Tao Sun, Jeronimo Castrillon, Kanishkan Vadivel, and Guangzhi Tang. Spiking and event-driven neuromorphic mamba models for efficient speech recognition. *arXiv preprint arXiv:2606.01135*, 2026.

Malyaban Bal and Abhronil Sengupta. P-spikessm: Harnessing probabilistic spiking state space models for long-range dependency tasks. In *Proc. ICLR*, volume 2025, pp. 73912–73927, 2025.

Andrea Banino, Jan Balaguer, and Charles Blundell. Pondernet: Learning to ponder. *arXiv preprint arXiv:2107.05407*, 2021.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Hannah Bos and Dylan R Muir. Micro-power spoken keyword spotting on xylo audio 2. *arXiv preprint arXiv:2406.15112*, 2024.

Younes Bouhadjar and Emre Neftci. Sparse axonal and dendritic delays enable competitive snns for keyword classification. *arXiv preprint arXiv:2602.09746*, 2026.

Jiechen Chen, Sangwoo Park, and Osvaldo Simeone. Knowing when to stop: Delay-adaptive spiking neural network classifiers with reliability guarantees. *IEEE Journal of Selected Topics in Signal Processing*, 19(1):88–102, 2025.

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.

Manon Dampfhoffer, Thomas Mesquida, Emmanuel Hardy, Alexandre Valentian, and Lorena Anghel. Leveraging sparsity with spiking recurrent neural networks for energy-efficient keyword spotting. In *Proc. ICASSP*, pp. 1–5. IEEE, 2023.

Manon Dampfhoffer, Thomas Mesquida, Alexandre Valentian, and Lorena Anghel. Backpropagation-based learning techniques for deep spiking neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 35(9):11906–11921, 2024.

Luke Darlow, Ciaran Regan, Sebastian Risi, Jeffrey Seely, and Llion Jones. Continuous thought machines. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Proc. NeurIPS*, volume 38, Main Conference, pp. 21548–21594. Curran Associates, Inc., 2025. doi: 10.52202/085713-0728.

Lucas Deckers, Laurens Van Damme, Werner Van Leekwijck, Ing Jyh Tsang, and Steven Latré. Co-learning synaptic delays, weights and adaptation in spiking neural networks. *Frontiers in Neuroscience*, 18:1360300, 2024. doi: 10.3389/fnins.2024.1360300.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. In *Proc. ICLR*, 2019.

Maxime Fabre, Lyubov Dudchenko, Younes Bouhadjar, and Emre Neftci. SiLIF: Structured state space model dynamics and parametrization for spiking neural networks. *arXiv preprint arXiv:2506.06374*, 2025. URL https://arxiv.org/abs/2506.06374v4.

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 *Proc. ICCV*, pp. 2641–2651. IEEE, 2021.

Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

Ilyass Hammouamri, Ismail Khalfaoui-Hassani, and Timothée Masquelier. Learning delays in spiking neural networks using dilated convolutions with learnable spacings. In *Proc. ICLR*, 2024. URL https://arxiv.org/abs/2306.17670.

Yangfan Hu, Qian Zheng, Guoqi Li, Huajin Tang, and Gang Pan. Toward large-scale spiking neural networks: A comprehensive survey and future directions. *arXiv preprint arXiv:2409.02111*, 2024.

Sarthak Jain and Byron C Wallace. Attention is not explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 3543–3556, 2019.

Chen Li, Edward G Jones, and Steve Furber. Unleashing the potential of spiking neural networks with dynamic confidence. In *Proc. ICCV*, pp. 13304–13314. IEEE, 2023a.

Xiaohuan Li, Yi Liu, and Libo Zheng. A lightweight keyword spotting method using a convolutional spiking neural network with learnable synaptic delays. *Sensors*, 26(14):4514, 2026.

Yuhang Li, Tamar Geller, Youngeun Kim, and Priyadarshini Panda. Seenn: Towards temporal spiking early exit neural networks. *Proc. NeurIPS*, 36:63327–63342, 2023b.

Yuhang Li, Abhishek Moitra, Tamar Geller, and Priyadarshini Panda. Input-aware dynamic timestep spiking neural networks for efficient in-memory computing. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6. IEEE, 2023c.

Zeyu Liu, Gourav Datta, Anni Li, and Peter Beerel. Lmuformer: Low complexity yet powerful spiking model with legendre memory units. In *Proc. ICLR*, volume 2024, pp. 12108–12122, 2024.

Thomas Louis, Alain Pegatoquet, Benoit Miramond, and Adrien Girard. On reducing activity with distillation and regularization for energy efficient spiking neural networks. In *Proc. ICANN*, pp. 407–421. Springer, 2024.

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

Prajna G Malettira, Shubham Negi, Wachirawit Ponghiran, and Kaushik Roy. Tskips: Efficiency through explicit temporal delay connections in spiking neural networks. *Transactions on Machine Learning Research*, 2025.

Balázs Mészáros, James C Knight, and Thomas Nowotny. Efficient event-based delay learning in spiking neural networks. *Nature Communications*, 16(1):10422, 2025.

Mashrur M Morshed and Ahmad Omar Ahsan. Attention-free keyword spotting. *arXiv preprint arXiv:2110.07749*, 2021.

Mattias Nilsson, Ton Juny Pina, Lyes Khacef, Foteini Liwicki, Elisabetta Chicca, and Fredrik Sandin. A comparison of temporal encoders for neuromorphic keyword spotting with few neurons. In *Proc. IJCNN*, pp. 1–7. IEEE, 2023.

João D Nunes, Marcelo Carvalho, Diogo Carneiro, and Jaime S Cardoso. Spiking neural networks: A survey. *IEEE access*, 10:60738–60764, 2022.

Parth Patne, Mahdi Taheri, Ali Mahani, Maksim Jenihhin, Reza Mahani, and Christian Herglotz. Sparq: Spiking early-exit neural networks for energy-efficient edge ai. *arXiv preprint arXiv:2603.14380*, 2026.

Alejandro Pequeño-Zurro, Lyes Khacef, Stefano Panzeri, and Elisabetta Chicca. Towards efficient keyword spotting using spike-based time difference encoders. *arXiv preprint arXiv:2503.15402*, 2025.

Alexandre Queant, Ulysse Rançon, Benoit R Cottereau, and Timothée Masquelier. Delrec: learning delays in recurrent spiking neural networks. *arXiv preprint arXiv:2509.24852*, 2025.

H Sabbella, A Mukherjee, T Kandappu, S Dey, A Pal, A Misra, and D Ma. The promise of spiking neural networks for ubiquitous computing: A survey and new perspectives. *arXiv preprint arXiv:2506.01737*, 2025.

Sofia Serrano and Noah A Smith. Is attention interpretable? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 2931–2951, 2019.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *Proc. ICLR*, 2017.

Yoonsik Shim, Andrew Philippides, Kevin Staras, and Phil Husbands. Unsupervised learning in an ensemble of spiking neural networks mediated by itdp. *PLoS computational biology*, 12(10): e1005137, 2016.

Zeyang Song, Qianhui Liu, Qu Yang, Yizhou Peng, and Haizhou Li. ED-sKWS: Early-Decision Spiking Neural Networks for Rapid, and Energy-Efficient Keyword Spotting. In *Proc. Interspeech*, pp. 4528–4532, 2024a.

Zeyang Song, Jibin Wu, Malu Zhang, Mike Zheng Shou, and Haizhou Li. Spiking-leaf: A learnable auditory front-end for spiking neural networks. In *Proc. ICASSP*, pp. 226–230. IEEE, 2024b.

Pengfei Sun, Jibin Wu, Paul Devos, and Dick Botteldooren. Towards parameter-free attentional spiking neural networks. *Neural Networks*, 185:107154, 2025.

Dewei Wang, Sung Justin Kim, Minhao Yang, Aurel A Lazar, and Mingoo Seok. Background noise and process-variation-tolerant sub-microwatt keyword spotting hardware featuring spike-domain division-based energy normalization. *IEEE Journal of Solid-State Circuits*, 60(2):685–694, 2025.

Jiaqi Wang, Liutao Yu, Liwei Huang, Chenlin Zhou, Han Zhang, Zhenxi Song, Honghai Liu, Min Zhang, Zhengyu Ma, and Zhiguo Zhang. Efficient speech command recognition leveraging spiking neural networks and progressive time-scaled curriculum distillation. *Neural Networks*, 195:108253, 2026a.

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 *Proc. AAAI*, volume 40, pp. 2119–2127, 2026b.

Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

Sarah Wiegreffe and Yuval Pinter. Attention is not not explanation. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 11–20, 2019.

Yukai Yang, Chenxi Qin, Jungang Li, Xin Zhang, Wenwei Shao, and Liqun Chen. Spiking-moe: Sdprompt-guided dynamic expert fusion in spiking neural networks. *arXiv preprint arXiv:2605.23188*, 2026.

Man Yao, Jiakui Hu, Zhaokun Zhou, Li Yuan, Yonghong Tian, Bo Xu, and Guoqi Li. Spike-driven transformer. *Proc. NeurIPS*, 36:64043–64058, 2023.

Sidi Yaya Arnaud Yarga and Sean U N Wood. Neuromorphic Keyword Spotting with Pulse Density Modulation MEMS Microphones. In *Proc. Interspeech*, pp. 3275–3279, 2024.

Jilin Zhang, Dexuan Huo, Jian Zhang, Chunqi Qian, Qi Liu, Liyang Pan, Zhihua Wang, Ning Qiao, Kea-Tiong Tang, and Hong Chen. Anp-i: A 28-nm 1.5-pj/sop asynchronous spiking neural network processor enabling sub-0.1-μ j/sample on-chip learning for edge-ai applications. *IEEE Journal of Solid-State Circuits*, 59(8):2717–2729, 2024.

Zhaokun Zhou, Yuesheng Zhu, Chao He, Yaowei Wang, Shuicheng Yan, Yonghong Tian, and Li Yuan. Spikformer: When spiking neural network meets transformer. In *Proc. ICLR*, 2023.

Zhaokun Zhou, Yijie Lu, Kaiwei Che, Wei Fang, Keyu Tian, Qihao Peng, Yuesheng Zhu, Shuicheng Yan, Yonghong Tian, and Li Yuan. Spikformer v2: Join the high accuracy club on imagenet with an snn ticket. *arXiv preprint arXiv:2401.02020*, 2024.

AI USE DISCLOSURE

We used generative AI tools to improve the readability and language of selected parts of the manuscript and to assist with the editing of figure labels and captions. We reviewed all AI-assisted text and visual edits manually and verified them against the implementation, experiments, and cited literature. The authors take full responsibility for the final content of this paper, including all claims, results, and artifacts.

SUPPLEMENTARY MATERIAL

A REPRODUCIBILITY AND ROBUSTNESS

A.1 TASK CONFIGURATIONS AND INPUT CONSTRUCTION

Table 5 records the settings for each task used for the reported models. Architecture width is fixed across tasks. Only recurrent depth, the number of adapter stages, optimizer settings for each task, and the refinement-loss switch differ. The seed used in the main paper, 312, is also one of the five robustness seeds.

Table 5: Frozen task configurations. LR and WD denote the initial learning rate and weight decay, respectively.

Data	Classes	D	Adapters	Epochs	LR	WD	Batch	λ_{ref}
SHD	20	2	1	500	0.010	0.010	256	0.00
SSC	35	8	4	300	0.005	0.020	256	0.05
GSC	35	4	2	300	0.002	0.005	256	0.05

The pooled token passes through a linear projection from d_T to d_s and LayerNorm, then a second linear map and LayerNorm at width d_s. A learned seed normalized across state dimensions is added before the shared state PLIF produces s_0.

All tasks use token/state/hidden dimensions $128/96/192$, eight reader heads, rank-8 adapters with two steps per stage, adaptive aggregation initialized at 0.05, hSEM dropout 0.15, AdamW, and cosine annealing with period 40. Dropout is applied after each hSEM spike layer, and dropout in the CTU step loop is zero. Teacher, logit distillation, token-relation distillation, distillation from the final step, the auxiliary head that averages token predictions, and flip-loss branches are disabled. All LIF and PLIF states are reset after every batch. Thus, Table 5 shows that model width and the recurrent mechanism are fixed across tasks. The main task-specific architectural variable is recurrent depth, with the number of adapter stages adjusted accordingly. The refinement switch is the only loss difference because the controlled SHD study finds it detrimental at $D = 2$.

SHD and SSC use 10-ms event integration at the reported $\mathsf{T} = 100$ setting. Cochlear indices are grouped in contiguous sets of five, reducing 700 sensors to 140 feature channels. Padding is zero and excluded by mask from pooling, attention, and token voting. Independent temporal and feature masks are sampled with probability 0.5. SHD masks at most 20% of the time steps and 20 feature channels, while SSC uses 10% of the time steps and 10 feature channels. GSC truncates or pads waveforms before the released 8-kHz log-Mel pipeline, uses $n_{\text{fft}} = 256$, a hop length of 80, and 140 Mel bins, then applies one frequency mask and one time mask. We preserve the released $f_{\max} = 14{,}000$ setting despite the 8-kHz target sample rate to avoid silently changing the benchmark feature pipeline.

A.2 TRAINING LOSSES AND POLICY CALIBRATION

For reproducibility, let $\varepsilon_{n,t} = \mathrm{CE}(\ell_{n,t}, y_n)$ denote per-sample cross-entropy. Late anytime weights allocate total mass 0.1 uniformly to the first $d = \lfloor D/2 \rfloor$ steps and mass 0.9 to the remaining steps in proportion to $v_i = \min(i, 3)$. Thus, for $D > 1$,

$$\omega_t = \begin{cases} 0.1/d, & t \le d, \\ 0.9 v_{t-d} / \sum_{i=1}^{D-d} v_i, & t > d, \end{cases} \qquad \sum_t \omega_t = 1, \tag{7}$$

while $\omega_1 = 1$ when $D = 1$. The outer anytime coefficient is $\lambda_{\text{any}} = 0.1$.

Preservation of the first prediction operates only on samples predicted correctly at the first step. With the margin set to zero, it averages $[\varepsilon_{n,t} - \varepsilon_{n,1}]_+$ over those samples and $t > 1$; its coefficient is $\lambda_{\text{pre}} = 0.05$. Belief refinement partitions samples at each $t > 1$ by the correctness of step $t - 1$. It averages $\varepsilon_{n,t}$ over samples that were previously incorrect and $[\varepsilon_{n,t} - \varepsilon_{n,t-1}]_+$ over samples that were previously correct. It then sums these two subset means and averages the result across later steps. Its preservation multiplier is one and $\lambda_{\text{ref}} = 0.05$ on SSC/GSC; SHD sets $\lambda_{\text{ref}} = 0$. An empty subset contributes zero, matching the released implementation.

SSC and GSC select thresholds using their full official validation splits. Because SHD has no official validation split, its policy for early exit instead uses a deterministic 20% subset of the training data, stratified by class, for calibration (seed 20260730). Thresholds are selected under at most 0.1 percentage-point loss on calibration relative to the adaptive output after all steps, without observing evaluation labels. Patience is one and the condition is strict, $q_t^{\mathrm{cert}} > h$.

A.3 Multi-seed robustness

Table 6: Five independent runs. Values are top-1 accuracy (%) under the official evaluation protocol; boldface and underlining indicate the best and second-best results, respectively, for each task. Tied second-best values are underlined.

Data	Params (M)	Seed 312	Seed 326	Seed 12306	Seed 11408	Seed 42	Mean ± s.d.
SHD	0.16	**96.60**	<u>95.80</u>	95.27	95.10	94.79	95.51 ± 0.71
SSC	0.17	**81.74**	<u>81.42</u>	81.38	81.25	81.18	81.39 ± 0.22
GSC	0.17	**95.96**	<u>95.80</u>	95.76	95.60	<u>95.80</u>	95.78 ± 0.13

The five-run means are $95.51 \pm 0.71\%$, $81.39 \pm 0.22\%$, and $95.78 \pm 0.13\%$ for SHD, SSC, and GSC, where the uncertainty is the sample standard deviation across seeds. Cross-seed ranges are 1.81, 0.56, and 0.35 points, respectively. Seed 312 exceeds the corresponding mean by 1.09 points on SHD, compared with 0.35 on SSC and 0.18 on GSC. Thus, the headline SHD result is more seed-sensitive, whereas the SSC/GSC models form tighter clusters around their means. SSC/GSC entries are independent-test evaluations of checkpoints selected on validation; SHD entries follow the official train/test reporting protocol used by prior work. These five runs quantify robustness; the single-seed depth, timestep, and ablation sweeps remain sensitivity evidence, not significance tests.

A.4 Baseline sources and reporting conventions

DCLS-Delays uses 2L-1KC on SHD and 3L-2KC on SSC/GSC, the most accurate configurations in Table 2 of Hammouamri et al. (2024). The d-cAdLIF results come from Table 3 of Deckers et al. (2024). For both methods, Table 1 lists the published mean accuracies without uncertainty intervals, not the best individual runs. Parameter counts are rounded to two decimal places. Both use 10-ms spiking input bins, but GSC preprocessing differs (140 versus 40 Mel bins). Neither supplies comparable energy estimates for these configurations. Energy values for the remaining baselines follow Tables 2–3 of Wang et al. (2026b).

B Sensitivity and Adaptive Computation

B.1 Sensitivity to recurrent depth

Table 7: Recurrent-depth sensitivity (top-1 accuracy, %). Boldface and underlining indicate the best and second-best results, respectively, for each task.

Data	$D = 1$	$D = 2$	$D = 4$	$D = 8$	$D = 16$
SHD	<u>95.49</u>	**96.60**	95.36	95.05	92.45
SSC	79.98	80.68	81.22	**81.74**	<u>81.46</u>
GSC	95.47	95.51	**95.96**	<u>95.94</u>	95.43

The full numerical results in Table 7 correspond to Fig. 6. Relative to $D = 1$, the task-specific optima improve SHD, SSC, and GSC by 1.11, 1.76, and 0.49 points. Unrolling to $D = 16$ instead loses 4.15, 0.28, and 0.53 points relative to those optima. The pronounced SHD decline, together with the shallower optima on SHD/GSC, shows that depth is not a uniform capacity multiplier. Each configuration is trained independently: a $D = 1$ model is not obtained by truncating a $D = 8$ model. Table 7 therefore measures the effects of training at each depth and recurrent dynamics, rather than optional post-training unrolling of an invariant predictor.

B.2 SENSITIVITY TO SENSORY TIME STEPS

SHD and GSC exhibit a saturating resolution regime, whereas SSC continues to benefit from finer temporal sampling. SHD rises from 87.94% at $\mathsf{T} = 10$ to 96.60% at $\mathsf{T} = 100$, then falls by 2.17 points at $\mathsf{T} = 200$. GSC similarly improves from 85.31% to 95.96% at $\mathsf{T} = 100$ before declining to 95.44% at $\mathsf{T} = 250$. In contrast, SSC rises from 66.75% to 82.53% across the sweep; its $\mathsf{T} = 250$ optimum is 0.79 points above the common-resolution $\mathsf{T} = 100$ result. Fig. 7 therefore supports task-specific sensory resolution rather than a monotonic cross-task rule. SHD and SSC use integration windows of 100, 50, 25, 20, 10, 5, and 4 ms, corresponding to 10, 20, 40, 50, 100, 200, and 250 nominal steps. GSC instead fixes the spectrogram window at 256 samples and varies hop length. Thus, T changes sensory evidence granularity before recurrent CTU inference rather than changing the number of steps.

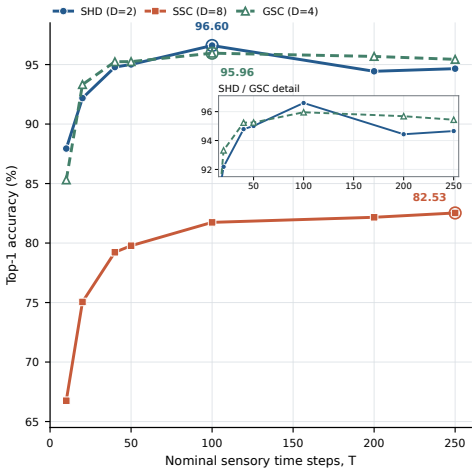

Figure 7: Sensory time-step sensitivity. Open markers show task-specific optima.

Beyond $\mathsf{T} = 100$, SHD and GSC decline while SSC continues to improve through $\mathsf{T} = 250$. Temporal resolution is therefore task dependent rather than a universally monotonic compute dimension. Main comparisons nevertheless retain $\mathsf{T} = 100$ so that input resolution and operation accounting remain fixed across tasks.

Table 8: Sensory time-step sensitivity (top-1 accuracy, %). Boldface and underlining indicate the best and second-best results, respectively, for each task.

Data	$\mathsf{T} = 10$	$\mathsf{T} = 20$	$\mathsf{T} = 40$	$\mathsf{T} = 50$	$\mathsf{T} = 100$	$\mathsf{T} = 200$	$\mathsf{T} = 250$
SHD	87.94	92.18	94.79	<u>95.01</u>	**96.60**	94.43	94.66
SSC	66.75	75.05	79.23	79.78	81.74	<u>82.17</u>	**82.53**
GSC	85.31	93.32	95.24	95.25	**95.96**	<u>95.68</u>	95.44

Table 8 provides the exact checkpoints underlying the compact plot. The different behavior at higher resolution is not compared vertically across datasets because class spaces, recurrent depths, and evaluation protocols differ. Every entry is one controlled run, so the sweep characterizes sensitivity rather than uncertainty.

B.3 ADAPTIVE COMPUTATION AND DIFFICULTY

Table 9: Difficulty-aware allocation. Tertiles of normalized Step 1 entropy define the easy, medium, and hard groups; depths that do not decrease across these groups are selected on calibration data subject to the primary budget for early exit.

Data	Depths (easy/medium/hard)	Accuracy (%)	Mean steps
SHD	1/1/1	95.63	1.00
SSC	1/3/5	81.12	3.06
GSC	1/1/2	95.61	1.35

Difficulty-aware allocation partitions calibration samples into tertiles by normalized Step 1 entropy and exhaustively chooses depths that do not decrease from the easy group to the hard group under the corresponding budget on the mean number of steps of the primary policy for early exit. Table 9 assigns depths $1/3/5$ on SSC and achieves 81.12% with a mean of 3.06 steps, compared with 81.42% at 3.28 steps for confidence stopping. On GSC, the coarse $1/1/2$ allocation achieves 95.61% with

a mean of 1.35 steps, whereas confidence stopping reaches 95.99% with a slightly lower mean of 1.32 steps. Step 1 entropy nevertheless correlates with the depth at which the confidence policy exits ($\rho = 0.8811$ on SSC and 0.6031 on GSC). Initial uncertainty is therefore informative, but an assignment to three bins discards sample-level variation used by the confidence policy.

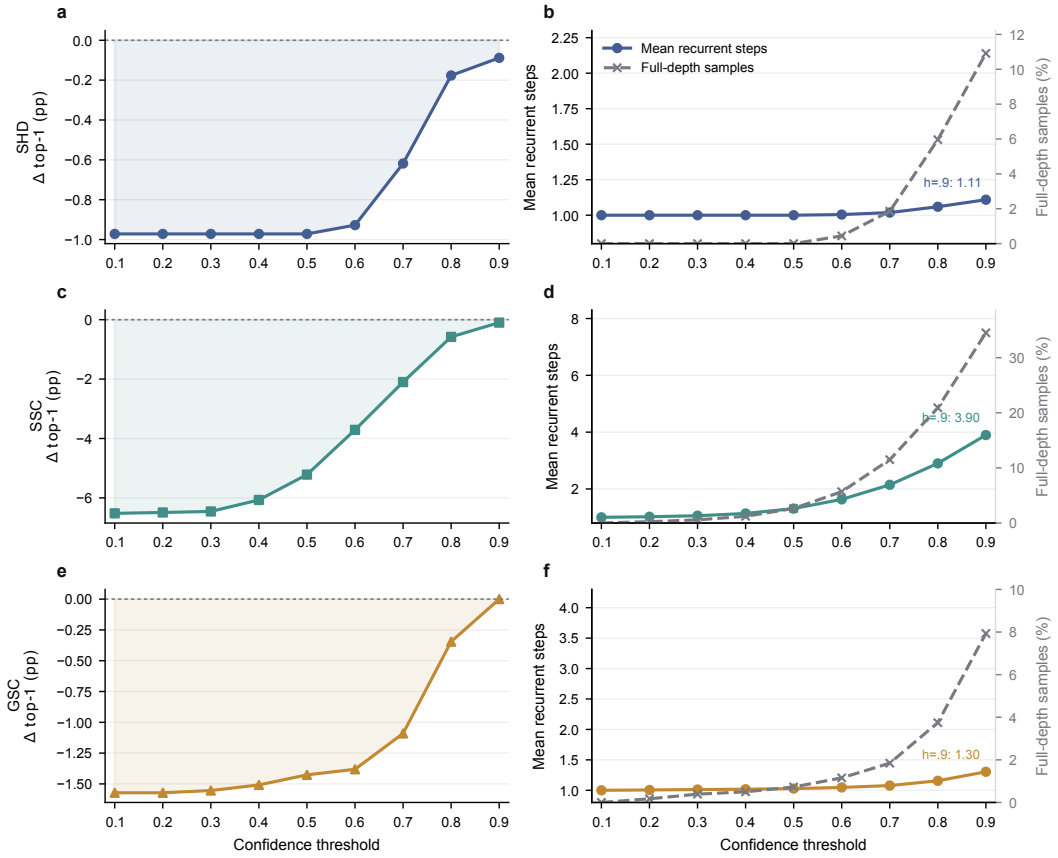

Figure 8: Complete confidence-threshold scan. Accuracy is relative to full adaptive inference; the right panels show the steps actually evaluated under each threshold and the fraction reaching full task depth. Reported thresholds are selected on calibration rather than from these evaluation curves.

The fixed-depth value used in the main paper linearly interpolates accuracy after a prefix of steps from the same frozen model selected for each task between adjacent integer prefix lengths at the dynamic policy's mean number of evaluated steps; it is not drawn from independently trained depth-search models. It is a descriptive control for comparison at the same average compute rather than a deployable fixed-depth system. The policy advantage on SSC and GSC indicates that confidence assigns recurrent steps more effectively than uniform truncation at the same average per-sample compute.

Fig. 8 exposes the full, uncalibrated threshold frontier. At $h = 0.9$, SHD evaluates an average of 1.11 steps out of 2 and remains 0.09 points below full inference; SSC evaluates an average of 3.90 steps out of 8, with 34.5% of samples reaching the full task depth, and remains 0.10 points below full inference; GSC evaluates an average of 1.30 steps out of 4, with only 7.9% reaching full depth, while matching full accuracy. These evaluation curves explain why a single numerical threshold is not transferable across tasks. Reported policies instead use validation calibration and the tolerance defined in Section A.

C Recurrent Dynamics and Temporal Evidence

C.1 Evolution of the binary spike state and prediction trajectories

For Fig. 4, trajectories initially wrong but ending correctly account for 34/2,264 SHD samples (1.50%), 1,933/20,382 SSC samples (9.48%), and 265/11,005 GSC samples (2.41%). Among these corrected samples, delayed corrections account for 35.08% on SSC and 24.53% on GSC, while oscillatory corrections account for 11.28% (218 samples) and 6.04% (16 samples), respectively. SHD uses only two recurrent steps, so repeated changes in correctness cannot occur. Conversely, 0.53%, 3.06%, and 0.84% of SHD, SSC, and GSC samples begin correctly but end incorrectly. These statistics follow successive predictions of the same trained model. For example, the first step of the SSC model trained at $D = 8$ achieves 75.22%, whereas a separate model trained at $D = 1$ achieves 79.98%.

Fig. 9 rules out a static-state interpretation of the shared CTU. SHD state firing rate rises from 29.76% to 41.00% over two steps, with 14.14% of state bits changing during the update. SSC firing rises from 23.60% to 43.89%; after the larger change between Step 1 and Step 2 (18.15%), consecutive flip rates remain between 14.51% and 15.47%. GSC firing increases from 33.57% to 51.95%, while its flip rate decreases from 22.45% at Step 2 to 17.42% at Step 4. The nonzero pairwise Hamming distances further show that the binary state vectors differ across steps, even when the top-1 prediction is unchanged.

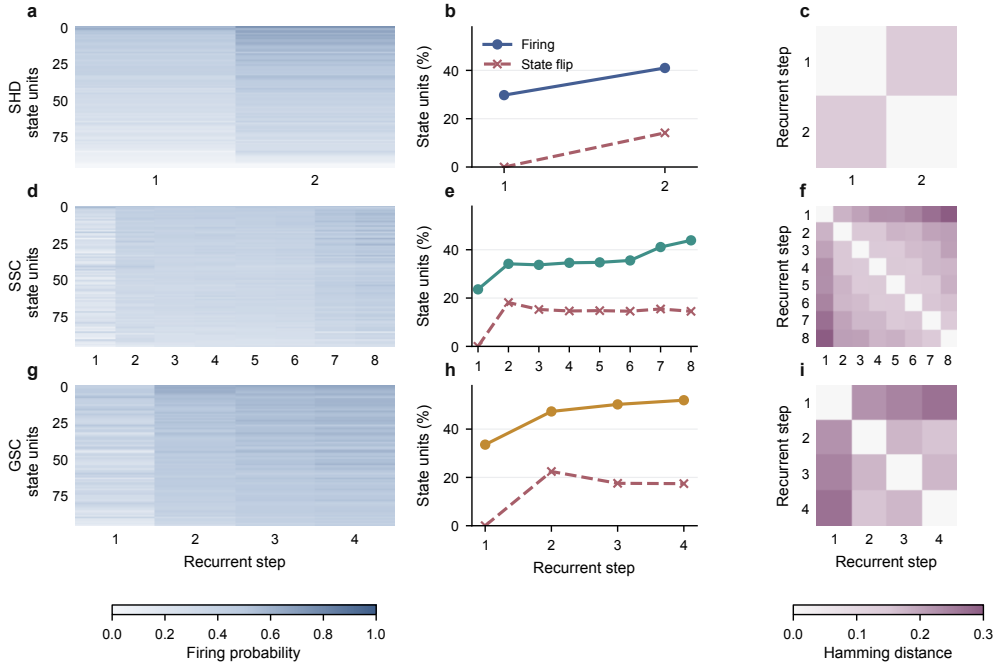

Figure 9: Dynamics of the recurrent binary spike state. Firing probabilities for each neuron and step, population firing and flip rates, and pairwise Hamming distances show that the binary state vectors evolve across steps. Neurons are independently sorted per task for visualization.

The state evolution also separates rapid initial correction from continued recurrent processing. From Step 1 to 2, top-1 accuracy rises by 0.97 points on SHD, 4.05 on SSC, and 1.23 on GSC, coinciding with each task's largest early change in the binary state. SSC then gains another 2.37 points by Step 8 even though its later state-flip rate stays near 15%, while GSC gains another 0.35 points by Step 4 as its flip rate decreases. The pairwise Hamming panels show that nonadjacent steps become increasingly separated, particularly on SSC, so the later updates are neither identical replay nor a simple two-state oscillation. Their diminishing accuracy return despite persistent state motion motivates aggregation and stopping based on prediction evidence rather than state change alone.

Fig. 10 connects those changes in the population state to class evidence. The selected SSC path follows wrong–correct–wrong–correct–wrong–correct–correct–correct while the target-class probability rises from 0.160 to 0.751; the adaptive aggregator assigns 66.9% of its weight to the final two steps, whose predictions remain stable. The GSC example follows wrong–correct–wrong–correct and assigns 50.6% to the recovered final step. Examples that change from correct to wrong provide the complementary failure mode: later recurrent updates can reinforce a competing class after an initially correct decision. These are deterministic representatives of path morphology, not frequency evidence; Analysis of prediction trajectories finds at least one oscillatory path in 218 of the 1,933 samples that end correctly on SSC and 16 of the 265 samples that end correctly on GSC.

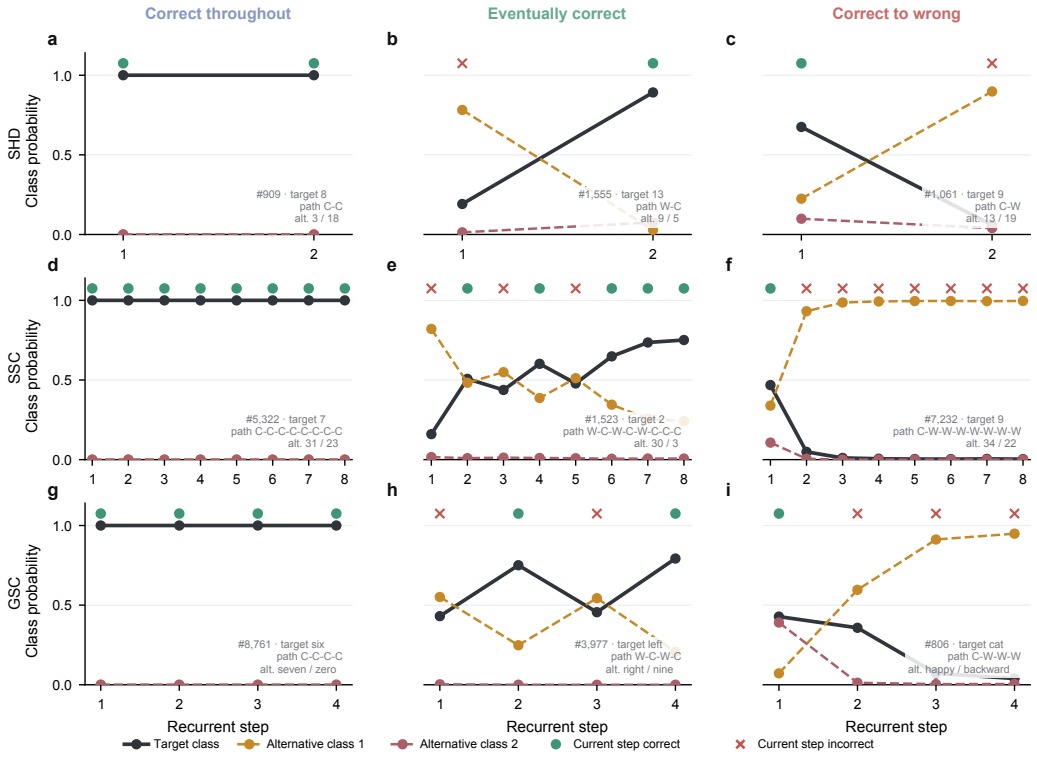

Figure 10: Representative trajectories for samples that remain correct, end correctly after initially being wrong, and change from correct to wrong. Examples illustrate path shape only; prevalence claims are based on the counts reported in Fig. 4.

C.2 Temporal attention across steps

Table 10: SSC attention migration and temporal input-occlusion results on the independent test set.

(a) Attention migration

Trajectory	Samples	JS div. ($\times 10^{-2}$)
Correct throughout	14,332	2.34
Wrong to correct	1,933	2.59
Correct to wrong	624	2.60

(b) Temporal occlusion

Occlusion	Accuracy (%)	Drop (pp)
Step 1 top 10%	63.29	18.45
Step 8 top 10%	64.09	17.65
Random 10%	77.19	4.55
Step 8 bottom 10%	81.16	0.57

We capture all eight reader-head attention distributions without changing logits. We measure attention migration using the Jensen–Shannon divergence between attention distributions at consecutive steps. The target margin is the target-class logit minus the largest non-target logit. Table 10 shows that trajectories with a correction and trajectories that degrade both show greater attention migration

than trajectories that remain correct: consecutive JS divergence rises from 0.02339 to 0.02588 and 0.02602, respectively. At the transition level, updates from wrong to correct have a JS divergence of 0.02785 and a flip rate in the binary state of 0.19142, compared with 0.02399 and 0.15217 when correctness is unchanged. Migration correlates with change in the binary state ($\rho = 0.3818$) but only weakly with immediate target margin change ($\rho = 0.0728$). Attention motion is therefore a marker of active belief revision, not evidence that the revision is necessarily beneficial.

Fig. 5 illustrates this distinction at the sample level. The example that remains correct maintains concentrated attention around its dominant event region and maintains high certainty. The corrected example reallocates mass across early and middle event segments before stabilizing on the target. In contrast, the degraded example broadens and shifts attention away from the initially dominant region; certainty subsequently recovers despite the wrong class. Recurrent re-reading can thus preserve, repair, or amplify a mistaken belief.

The three paths further separate certainty from correctness. In the corrected example, the prediction changes from wrong to correct at Step 2 as attention is redistributed over the early event region and a secondary region near the utterance middle; certainty then rises from roughly 0.65 toward one. In the degraded example, a sharp initial read near 400 ms supports the correct Step 1 decision, but later attention broadens, the prediction becomes wrong, and certainty nevertheless recovers to approximately 0.8–0.9. The stable sample, meanwhile, assigns most aggregation mass to later steps even though its class decision never changes. These contrasts explain why the aggregator conditions on recurrent state and step position in addition to certainty, and why certainty alone is insufficient as an interpretation of prediction quality.

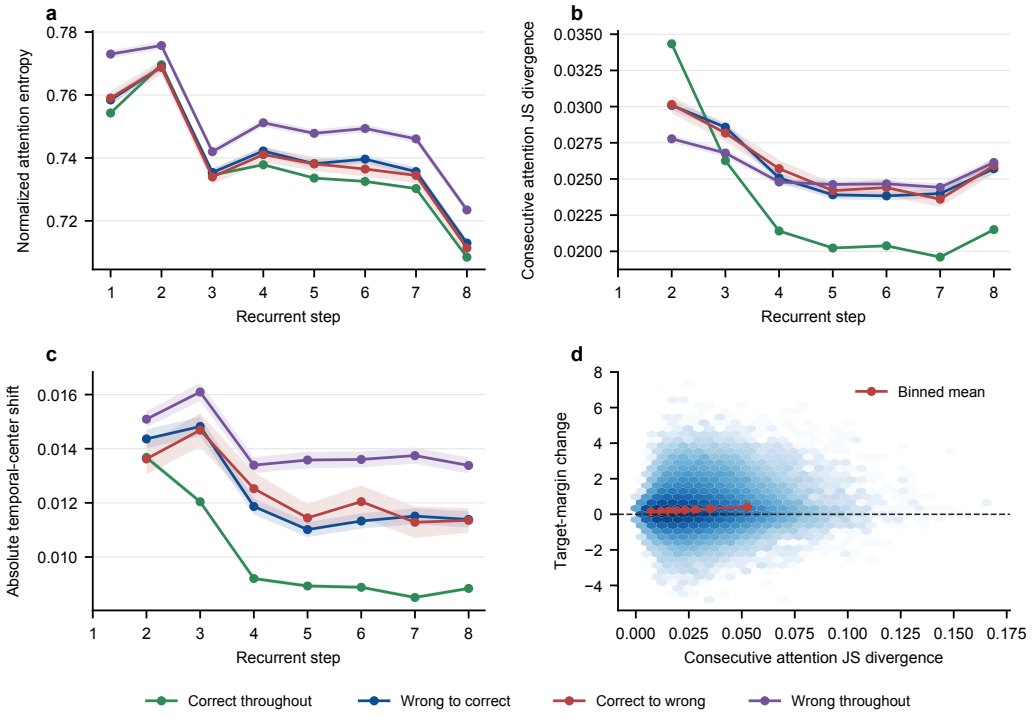

Figure 11: Population attention entropy, consecutive Jensen–Shannon divergence, center shift, and relation between migration and target margin change. Shading denotes standard error over test samples.

Fig. 11 confirms that migration is not confined to selected examples. Among 20,382 SSC test samples, 14,332 remain correct throughout, 1,933 are corrected, 624 begin correctly but end incorrectly, and the remaining 3,493 never reach the correct class. The corrected and degraded groups both exceed the JS level for trajectories that remain correct. The bootstrap intervals for corrected and stable

trajectories do not overlap. However, the weak $\rho = 0.0728$ relation to immediate margin change shows that the magnitude of attention movement alone does not determine the direction of the class update.

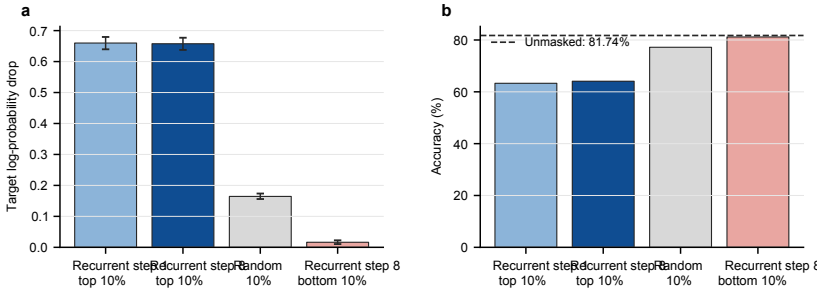

Figure 12: Temporal occlusion of the input. Masking the highest-attention bins is substantially more damaging than masking matched random bins or the lowest-attention bins; the dashed line is unmasked accuracy.

Table 10b and Fig. 12 test whether the attended input regions are decision-relevant rather than relying on attention weights alone. Masking the top 10% of bins attended at the final step reduces accuracy from 81.74% to 64.09%, compared with 77.19% under matched random masking and 81.16% after bottom-bin masking. The target-log-probability drop exceeds random masking by 0.4931 (bootstrap 95% interval $[0.4749, 0.5116]$) and bottom masking by 0.6412 (bootstrap 95% interval $[0.6209, 0.6611]$). Step 1 top-bin masking is similarly damaging: the paired target-log-probability difference (masking based on the final step minus Step 1 masking) is only -0.0023 (bootstrap 95% interval $[-0.0172, 0.0142]$), indicating no meaningful difference in damage between the two masking conditions. Attended regions are therefore decision relevant, but later steps do not monotonically discover better temporal locations; they dynamically reweight evidence that is already informative.

D Operation Proxies and Energy Boundaries

Table 11: Operation counts at the full depth selected for each task and 45-nm-process proxy energy at $\mathsf{T} = 100$.

Data	D	SOPs (G)	MACs (G)	Total ops (G)	Energy (mJ)
SHD	2	0.0014	0.0028	0.0041	0.0140
SSC	8	0.0016	0.0031	0.0047	0.0158
GSC	4	0.0006	0.0049	0.0055	0.0231

Table 12: Theoretical lower bound obtained by running one step; tokenizer and memory construction remain included.

Data	Task-selected full D	Accuracy (%)	SOPs (G)	Energy (mJ)	Energy reduction (%)
SHD	2	95.63	0.0013	0.0139	0.76%
SSC	8	75.22	0.0015	0.0150	4.90%
GSC	4	94.38	0.0005	0.0227	1.50%

For a spike-driven layer l, we estimate $\mathrm{SOP}_l = \mathrm{FR}_{l-1}\mathrm{FLOP}_l$ from its measured input firing rate; operations on continuous paths remain MACs. Proxy energy uses 0.9 pJ per accumulation operation and 4.6 pJ per multiply–accumulate. Table 11 shows that GSC has the lowest SOP count (0.0006 G) but the highest proxy energy (0.0231 mJ), because its 0.0049 G continuous MAC path dominates both total operations and the higher per-operation energy coefficient. SHD and SSC have more SOPs but smaller MAC totals, yielding lower proxy energy. Sparse-operation count alone is therefore insufficient for comparing hybrid models.

Table 12 gives the theoretical lower bound obtained by running one step while retaining tokenizer and memory construction. Relative to depth selected for each task, it lowers proxy energy by only 0.76% on SHD, 4.90% on SSC, and 1.50% on GSC, while reducing accuracy by 0.97, 6.52, and 1.58 points. The limited energy change shows that most estimated cost is paid before recurrent depth can be shortened; SSC has the largest recurrent share and consequently the largest energy reduction from running one step, but also the largest accuracy loss. Neither table includes memory movement, nonlinear functions, normalization, spike-neuron updates, control, or device utilization, and neither replaces measurement on a specified hardware platform. For spike-form inputs (SHD, SSC), SpikCommander's convention treats all operations as ACs (Wang et al., 2026b); we instead count continuous residual paths as MACs, so part of the SHD energy difference (0.0140 versus 0.0039 mJ) reflects accounting conventions rather than computational work. As in recent spiking speech work, theoretical 45-nm-process proxy energy is a common field-standard estimate in the absence of device access; deployment and measurement on neuromorphic hardware remain future work.