SPIKING NEURAL NETWORKS INCREMENTALLY COMPUTE CONTINUOUS-VALUED NEURAL NETWORKS THROUGH ATTRACTOR DYNAMICS

Anonymous authors
Paper under double-blind review

ABSTRACT

Understanding how spiking neural networks (SNN) efficiently process data streams is a key to (i) study how biological brains conserve energy for lifelong predictions and (ii) enable an efficient continuous AI inference. In this work, we propose a theory on how an SNN incrementally computes its corresponding continuous-valued neural network (ANN), namely by reusing prior activation and updating only the temporal difference between ANN and the SNN. Specifically, we prove that discrete SNN output continuously tracks the instantaneous ANN output implicitly determined by the SNN input. Based on this principle, we develop LIF-Stream, a efficient streaming SNN inference framework which controls the SNN to continuously adapt to the reference ANN output which computes the high-accuracy frame-wise prediction online. Evaluation on three large-scale streaming inference benchmarks with distinct modalities show that LIF-Stream achieves the best efficiency-accuracy trade-off compared to high-accuracy SNN techniques.

1 INTRODUCTION

Understanding how biological brains efficiently process continuous sensory streams throughout their lifetimes can advance the design of efficient, always-on AI systems. Brains save metabolic energy on static scenes while selectively scaling cognitive compute on surprising events (Attwell & Laughlin, 2001; Friston, 2010). Spiking neural networks (SNN) computationally model this biological neural systems to (i) study their underlying information processing mechanisms and (ii) implement extremely energy-efficient AI systems with neuromorphic hardwares (Roy et al., 2019). SNNs are networks of spiking neurons that mimic neuronal dynamics and communicate through binary spikes (Maass, 1996b; Gerstner & Kistler, 2002). Since they fundamentally process the time series of binary events (spike trains) in a biophysical manner, they are naturally expected to process streaming data efficiently. However, existing high-accuracy SNN techniques, including direct SNN training and ANN-to-SNN conversion methods, overlook the need to leverage historical states to efficiently process new inputs, especially for unbounded streams.

We first derive a theoretical principle governing the transient dynamics of discrete SNNs, which underlies how an SNN can efficiently process information through time. It is challenging to analyze the discrete SNNs since they are discontinuous dynamical systems. We overcome this by extending the ANN-to-SNN conversion theory (Rueckauer et al., 2016; Xiao et al., 2021; Oh & Lee, 2024) to the transient regimes through the lens of attractor dynamics. Specifically, we prove that SNN consisting of discrete LIF neurons (LIF-SNN) continuously attracts toward the instantaneous output of its corresponding continuous-valued clipped-ReLU network (ANN). Thus, the ANN output is a time-varying reference target that SNN dynamics tracks. LIF-SNN dynamics is a damped fixed-point iteration with a constant step size that permanently tracks this attractor. For static inputs, it oscillates within the small neighborhood of the static attractor. SNNs with discrete Integrate-and-Fire (IF) neurons also yield similar theoretical results, but their step sizes decay through time.

Based on our theory, we introduce *LIF-Stream*, an efficient streaming SNN inference framework in which the SNN incrementally computes its continuous-valued ANN counterpart across an unbounded input stream. Specifically, LIF-Stream is a two-time-scale SNN control process: The outer loop continuously encodes the input stream to *implicitly* drive the reference ANN for accurate frame-wise prediction. The inner loop allocates a few time steps per frame for the SNN to continuously track

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.

this reference ANN output, which is a time-varying instantaneous attractor, while retaining its states across frame boundaries.

LIF-Stream efficiently processes streams, since SNN retains prior activation values and incrementally computes to close the gap between ANN and SNN activations. For temporally correlated input streams, SNNs cost few ticks per frame on average to maintain high frame-wise accuracy. This finding challenges the common belief by showing that ANN-to-SNN conversion can be as efficient as direct training methods on streaming data without specialized training or fine-tuning. For such incremental computation to persist indefinitely, we argue that ANN-to-SNN conversion techniques should pivot from IF to LIF neuron. While IF neuron's time-diminishing step size mandates periodical global reset, LIF neuron's constant step size ensures permanent and autonomous alertness.

We comprehensively evaluate our framework across diverse streaming inference tasks spanning vision, audio, and neuromorphic (event-stream) benchmarks. Comparisons with state-of-the-art SNN techniques show that LIF-Stream generally achieves the best efficiency-accuracy trade-off. LIF-Stream is significantly tick-efficient than conversion techniques, while surpassing directly trained SNNs in peak accuracy with comparable low-tick accuracy. Furthermore, LIF-Stream is natively compatible with standard digital neuromorphic hardware out-of-the-box, as it operates without post-training model calibration, customized neuronal dynamics, or non-binary spike communication. In-depth analyses demonstrate that this tick efficiency stems from our incremental computing approach, which naturally exploits small inter-frame prediction deltas by computing incremental state updates rather than full activations.

2 Computing Dynamics of Spiking Neural Networks

We study the standard Leaky Integrate-and-Fire (LIF) neuron model (Gerstner et al., 2014), which is widely researched and implemented in neuromorphic hardware. The discrete-time dynamics of an LIF neuron with membrane potential $u(t)$ and spike output $s(t)$ consists of three consecutive steps:

$$u_{\text{pre}}(t) = \left(1 - \frac{1}{\tau}\right) u(t-1) + \frac{1}{\tau}\, u_{\text{rest}} + \frac{R}{\tau}\, I(t) \qquad \text{(Integration)} \tag{1}$$

$$s(t) = \mathbb{H}(u_{\text{pre}}(t) - \theta) \qquad \text{(Thresholding)} \tag{2}$$

$$u(t) = u_{\text{pre}}(t) - \theta\, s(t) \qquad \text{(Reset)} \tag{3}$$

Here, $u_{\text{pre}}(t)$ and u_{rest} denote the pre-firing and resting potential, $\mathbb{H}(\cdot)$ is the Heaviside step function, θ is the threshold, τ is the leakage factor, and R is membrane resistance. We adopt a soft reset (equation 3), the subtraction-based discretization of neuronal reset. We choose it instead of hard reset for its superior accuracy and analytical tractability (Han & Roy, 2020).

Integrate-and-Fire (IF) neuron further simplifies the integration dynamics by removing the leakage behavior through $\tau \to \infty, R = \tau$.

We use two decoding schemes to map binary spike trains $s(t)$ into real-valued signals $y(t)$. First, the standard rate coding averages all spikes equally through time. Second, the exponential moving average (EMA) coding (Oh & Lee, 2024) averages spikes with weights that exponentially decays through time. For analytical tractability, we align the decay factor of EMA coding with the leakage factor τ of the LIF neuron. Both admit streaming formulation, starting from $y(0) = 0$:

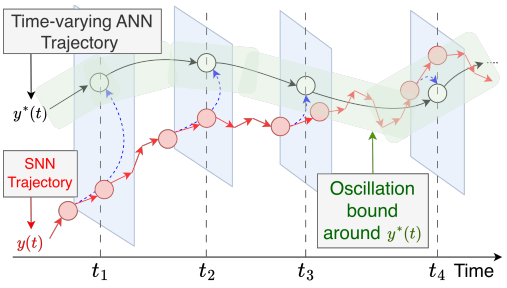
$$y(t+1) = \frac{\tau - 1}{\tau} y(t) + \frac{1}{\tau} s(t) \quad \big(\text{EMA}\big) \qquad\qquad y(t+1) = \frac{t-1}{t} y(t) + \frac{1}{t} s(t) \quad \big(\text{Rate}\big) \tag{4}$$

We consider a general fully connected network of d neurons driven by n input sources. An input matrix $F \in \mathbb{R}^{d \times n}$ weighs input signals $x(t)$, and a recurrent matrix $W \in \mathbb{R}^{d \times d}$ weighs previous spike outputs $s(t-1)$. Assuming linear synapse, the total current $I(t) \in \mathbb{R}^d$ is given by:

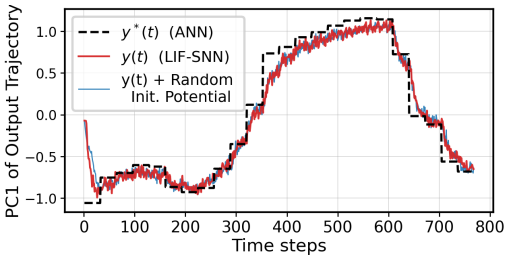
$$I(t) = Fx(t) + Ws(t-1). \tag{5}$$

2.1 Theoretical Analysis

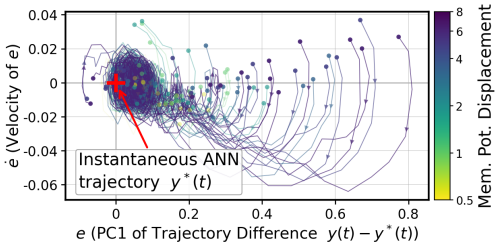
The LIF-SNN dynamics in equation 1-5 is a complex nonlinear system of difference equations that is difficult to analyze directly. We first show that this dynamics reduce to a single damped fixed-point



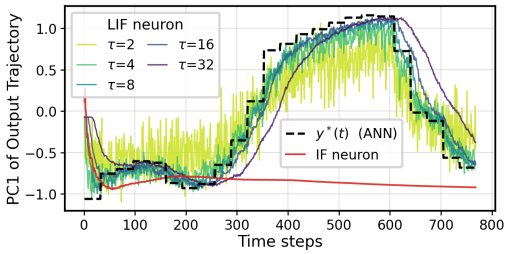
(a) Conceptual overview of our theory: LIF-SNN dynamics constantly tracks time-varying ANN outputs.



(b) Empirical validation of our theory: Time-domain tracking of LIF-SNN towards its ANN Trajectory.



(c) Multiple LIF-SNN trajectories starting from random initial conditions converge toward the time varying ANN trajectory given identical input sequences.



(d) Impact of step size on SNN tracking dynamics. Unlike constant $1/\tau$ (LIF neuron), time-decaying step size $1/(t+1)$ (IF neuron) fails to track continuously.

Figure 1: Conceptual overview and empirical validation of theoretical findings. (b) LIF-SNNs are continuously attracted to instantaneous ReLU1-ANN outputs, tracking the ANN trajectory through time. (c) Basin of attraction: trajectories starting from arbitrary initial states and histories converge toward the same target trajectory given identical inputs afterwards. (d) Trade-offs in step size (τ): larger step sizes (smaller τ) achieve faster convergence with larger oscillation, while smaller step sizes (larger τ) reduce oscillation at the expense of convergence speed.

iteration. Throughout the paper, we list complete proofs in Appendix C. $\mathbf{I}$ denotes the identity matrix, $\mathbf{1}$ the vector of ones, and ReLU1 : $\mathbb{R} \to [0, 1]$ defined as $\text{ReLU1}(x) = \min(\max(x, 0), 1)$.

Lemma 2.1. *Let the input signals $x(t)$ and spike outputs $s(t)$ be EMA-coded into $\tilde{x}(t)$ and $y(t)$, respectively. The output trajectory $y(t)$ of a discrete-time LIF-SNN exactly equals the trajectory of the damped fixed point iteration with a constant factor $\frac{1}{\tau}$:*

$$y(t) = \left(1 - \frac{1}{\tau}\right) y(t-1) + \frac{1}{\tau}\,\mathbb{H}\Big(F\,\tilde{x}(t) + W_0\,y(t-1) + b(t) - \theta \cdot \mathbf{1}\Big) \tag{6}$$

where $\alpha = 1 - \frac{1}{\tau}$, $W_0 \coloneqq W - (\tau - 1)\theta\mathbf{I}$, and $b(t) \coloneqq u_{rest}(1 - \alpha^t)\mathbf{1} + \alpha^t u_{pre}(0)$.

Proof sketch. We eliminate the membrane potentials $u(t), u_{\text{pre}}(t)$ in terms of $\tilde{x}(t)$ and $y(t)$. Recursively unrolling the integration equation 1 backwards to $t = 0$ expresses $u_{\text{pre}}(t)$ as an exponentially weighted sum of $I(k)$. Substituting $I(k)$ using equation 5 reduces $u_{\text{pre}}(t)$ to a linear combination of $y(t-1)$ and $\tilde{x}(t)$:

$$u_{\text{pre}}(t) = F\,\tilde{x}(t) + W_0\,y(t-1) + b(t), \tag{7}$$

Plugging this equation 7 and the firing rule equation 2 into the coding definition equation 4 yields the desired results (see complete proof in Appendix C.1). □

Based on this, we derive our key finding governing the computing direction of transient SNN dynamics: the SNN output $y(t)$ continuously tracks its corresponding continuous-valued ReLU1 network *instantaneously* determined by the input $\tilde{x}(t)$ at each time step t, as shown in Figure 1a.

Theorem 2.2. *For each time step t, the LIF-SNN output $y(t)$ is attracted with a step size $\frac{1}{\tau}$ toward the fixed-point set $\mathcal{E}(t)$ of a continuous-valued neural network, instantaneously defined by $\tilde{x}(t)$:*

$$\mathcal{E}(t) \coloneqq \left\{ y \in [0,1]^d \;\middle|\; y = \textit{ReLU1}\Big(\tfrac{1}{(\tau-1)\theta}\big(F\tilde{x}(t+1) + Wy + b(t+1) - \theta \cdot \mathbf{1}\big)\Big) \right\} \tag{8}$$

The set $\mathcal{E}(t)$ is always nonempty because its ReLU1 map is a continuous self-map on the compact convex domain $[0,1]^d$, guaranteeing the existence of a fixed point by Brouwer's fixed-point theorem.

Proof sketch. The update term equation 6 can be rewritten as $y(t+1) = y(t) - \frac{1}{\tau}(y(t) - \mathbb{H}(D(t;y(t)) - y(t)))$, where $D(t;y) \coloneqq \frac{1}{(\tau-1)\theta}\big(F\tilde{x}(t+1) + b(t+1) + Wy - \theta \cdot \mathbf{1}\big)$. A coordinate-wise analysis shows that each coordinate $[y(t)]_k$ always moves toward $\mathrm{ReLU1}([D(t;y(t))]_k)$ with a constant step size $\frac{1}{\tau}$. This contracts the gap between $[y(t)]_k$ and $\mathrm{ReLU1}([D(t;y(t))]_k)$ at a geometric rate $\alpha = 1 - \frac{1}{\tau}$, towards the set of points satisfying $y = \mathrm{ReLU1}(D(t;y))$, which is $\mathcal{E}(t)$. Hence, $y(t)$ is attracted towards the instantaneous fixed-point set $\mathcal{E}(t)$ regardless of the initial condition or prior history $y(<t)$. (See full proof in Appendix C.2.) □

$\mathcal{E}(t)$ can be identified as a Deep Equilibrium Model (DEQ) (Bai et al., 2019), which defines the output as the fixed point of a single transformation driven by input. A single-layer DEQ is equivalent to infinite-depth, weight-tied feedforward network, and thus can represent any finite-depth network.

The constant step size makes $y(t)$ continuously pursue the moving target $\mathcal{E}(t)$, but also prevents exact convergence. Using Theorem 2.2, we can derive a stronger version of ANN-to-SNN conversion than prior work (Xiao et al., 2021): When input stabilizes and weight is contractive, $y(t)$ converges geometrically into an $\mathcal{O}(\frac{1}{\tau})$-neighborhood of the static equilibrium and oscillates within it thereafter.

Corollary 2.3 (ANN-to-SNN Conversion)**.** *Let $\gamma \coloneqq \frac{\|W\|_\infty}{(\tau-1)\theta} < 1$. If $\tilde{x}(t) \to \mathbf{x}$, then $\mathcal{E}(t) \to \mathcal{E}^*$, i.e.,*

$$\mathcal{E}^* \coloneqq \big\{ y \in [0,1]^d \;\big|\; y = \textit{ReLU1}\big(\tfrac{1}{(\tau-1)\theta}(F\mathbf{x} + Wy + (u_{rest} - \theta)\mathbf{1})\big)\big\} \tag{9}$$

is a singleton $\{y^*\}$, and $y(t)$ approaches its $\frac{1}{\tau(1-\gamma)}$-neighborhood at the geometric rate $\max\{\alpha,\gamma\}$, remaining there thereafter: $\limsup_{t\to\infty} \|y(t) - y^*\|_\infty \le \frac{1}{\tau(1-\gamma)}$ (Proof at Appendix C.3.)

IF Neuron. Similar theoretical properties hold for IF neurons under rate coding (See Appendix C.4). IF-SNN dynamics simplify into a fixed-point iteration with a step size $\eta(t) = \frac{1}{t+1}$. The trajectory is attracted toward its corresponding ReLU1-DEQ model outputs $\mathcal{E}_{\text{IF}}(x(t))$. Unlike LIF-SNNs, the time-diminishing characteristic of the step size $\eta(t)$ (Robbins & Monro, 1951) makes the IF-SNN trajectory converge *exactly* towards $\mathcal{E}_{\text{IF}}(\mathbf{x})$ once the input stabilizes $x(t) \to \mathbf{x}$, under $\|W\|_\infty < \theta$.

$$\mathcal{E}_{\text{IF}}(x) = \big\{ y \in [0,1]^d \mid y = \mathrm{ReLU1}\big((Wy + Fx)/\theta\big)\big\} \tag{10}$$

This analytical mapping between the IF neuron and clipped ReLU at steady state (Rueckauer et al., 2017; Han & Roy, 2020) has been foundational for ANN-to-SNN conversion. For the first time, our findings reveal the principle of SNN transient dynamics underlying this steady-state property.

Empirical Validation. We validate our theoretical findings through a toy experiment. We train a 7-layer MLP with ReLU1 activations to regress a smooth multivariate sinusoidal function, and convert it to SNN by replacing ReLU1s with LIF (or IF) neurons. Figure 1 show that LIF-SNN output continuously tracks its corresponding ReLU1-ANN's trajectory through time with a broad basin of attraction. Interestingly, the tracking dynamics governed by step sizes closely mirror online optimization dynamics: larger step sizes accelerate convergence at the expense of higher oscillation, whereas smaller step sizes yield smoother yet lags behind. In contrast, time-diminishing step size (IF neuron) greatly lags behind the time-varying target (Detailed experimental setup is at Appendix I.5.)

3 Controlling SNNs to efficiently process data streams

In the last section, we discovered a core principle governing the transient dynamics of SNN. We now leverage this principle to establish a framework that processes data streams accurately and efficiently with SNNs. Specifically, Theorem 2.2 shows that SNN constantly updates its output a step towards its corresponding ANN output. We *control* the dynamics of SNN using this governing principle so that SNN continuously tracks its corresponding ANN as a reference model.

Our *LIF-Stream* framework is a two-time-scale control that continuously adapts SNN to its time-varying reference model. It consists of (i) an inner loop, which SNN output closely tracks the output of the reference ANN model, and (ii) an outer loop, which the reference ANN continuously evolves its output given input stream. To simply put, we continuously tune the reference ANN, which is easier to train and stably infers, to robustly control the unstable and discontinuous SNN computation.

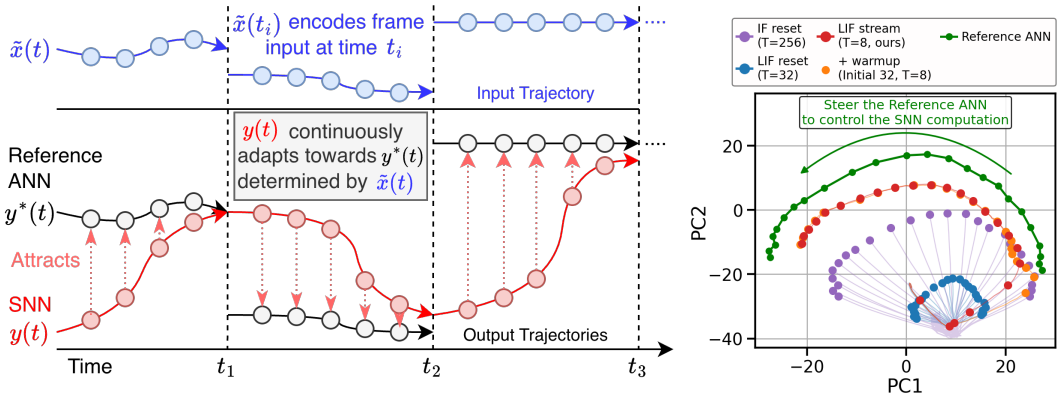


Figure 2: Conceptual overview and empirical trajectory analysis of **LIF-Stream**. (Left) Conceptual diagram of our LIF-Stream framework for streaming SNN inference. (Right) Empirical validation on MOT17 video sequences (Milan et al., 2016) evaluated as a streaming pedestrian detection task. LIF-Stream efficiently tracks the continuous ANN output by updating only inter-frame delta, whereas standard reset-based IF and LIF-SNNs suffer from re-convergence latency overhead for every frame.

Figure 2 (left) illustrates the concept of our framework. SNN continuously updates its output to adapt to a new reference ANN output. Here, the inner loop can be framed as an online adaptation process of SNN y to its reference model y^* in a time window of $t \in [T_{i-1}, T_i]$, as follows:

- Estimator: The SNN output $y(t)$ encoded in spike trains.
- Initial State: Prior SNN output $y(T_{i-1})$.
- Goal: The ANN output $y^*(t)$ that stably evolves towards desired frame output $y^*(T_i)$.
- Update Rule: The forward propagation of SNN, a fixed point iteration (Lemma 2.1.) that attracts $y(t)$ towards the reference ANN output $y^*(t)$.
- Step size: Constant $\frac{1}{\tau}$ (LIF) or time-decaying $\frac{1}{t+1}$(IF)

We pretrain a reference ReLU1-ANN with floating-point activations as a normal DNN. Unlike prior ANN-to-SNN conversion methods that utilizes IF neurons, we convert ReLU1 functions into LIF neurons based on the steady-state equivalence in Theorem 2.3. Since the reference ANN achieves high accuracy with standard gradient-based optimizations, the SNN could also achieve high streaming accuracy as long as it stably and closely tracks the reference model.

Algorithm 1 simply implements our framework with LIF-SNNs. Such simplicity comes from two aspects. First, the principle we discovered intrinsically links SNN with its reference ANN through inputs, without additional component. Second, the step size of LIF dynamics is constant throughout the adaptation process. These preclude any intervention during stream processing. As in Algorithms 2–7, input streams could be arbitrarily encoded, as long as the reference ANN stably and swiftly evolves towards the desired frame output. We empirically compare the encoding schemes in Appendix D.1.

Streaming Efficiency Our framework reveals how SNNs could be exceptionally efficient for processing temporally correlated data streams. We view SNN dynamics as an incremental computing algorithm of continuous-valued neural networks; It retains prior state and computes only the incremental updates in neural activations induced by input changes. For continuous, temporally correlated streams, the activations change gradually rather than abruptly. When the prior SNN state is already close to the target ANN output, and the reference ANN output changes minimally, the SNN requires only a small number of time steps to process a new frame. Hence, SNN naturally exploits the temporal redundancy without sacrificing accuracy, whereas conventional DNN inference should fully recompute regardless of temporal correlation.

Perpetual Inference Incremental computation is most effective when it can operate continuously without external reset or intervention. However, SNN inference systems frequently employ global reset that re-initialize all the neuronal states each frame to start with a clean slate. For our framework, this introduces three key drawbacks: (i) it completely discards the prior states that SNNs rely on for streaming efficiency, (ii) SNNs struggle to recover the accuracy after the reset, and (iii) it triggers global synchronization, which makes real-time systems less responsive to sudden events.

We observe that LIF neurons should be used instead of IF neurons to make neural computation perpetually incremental. Prior approaches using IF neuron (or its variants) are fundamentally limited

because IF-SNNs have a *time-diminishing step size*. The number of time steps (i.e., ticks) to track the reference model scales linearly over time, and thus it freezes after a few frames. Thus, IF-SNNs should globally reset periodically to maintain the streaming accuracy. While recent IF-based conversion techniques show high accuracy in ultra-low ticks, they are only effective in the first few ticks after global reset. In contrast, LIF neuron's *time-constant step size* makes the SNN permanently alert. LIF-SNN can continuously track the reference model's trajectory closely without any intervention. It is also naturally robust to long time horizon and distributional shifts in input streams.

Initial Wakeup. SNNs can exhibit a cold start phenomenon: a temporary initial accuracy drop where the SNN output $y(t)$ requires several ticks to warm up from $y(0)$ to match the reference target $y^*(T_1)$. This can be problematic in real-world scenarios when an agent should wake up fast and react. To resolve this, we dynamically schedule more time-steps solely for the first frame, e.g., 32 ticks in Figure 2 (right). This design preserves both initial accuracy and a low average-case tick budget.

Empirical validation on static-camera video streaming perception. Figure 2 (right) demonstrates our framework on MOT17 video sequences (Milan et al., 2016) repurposed for streaming pedestrian detection. We use an FCOS-lite architecture (Tian et al., 2019) with a VGG backbone (Simonyan & Zisserman, 2014) restricted to affine layers and ReLU1, trained on COCO (Lin et al., 2014b). We project the SNN output trajectories onto a 2D PCA subspace fitted to the ANN trajectory. LIF-Stream smoothly tracks the reference ANN via local incremental updates within 8 ticks. In contrast, reset-based baselines reinitialize to the origin each frame, and thus incur significant time steps. Also, initial warm-up aligns LIF-Stream with the ANN output in earlier frames.

4 Evaluation

To demonstrate generalizability, we extensively evaluate our framework across three distinct stream processing modalities, including vision, audio, and event streams. We benchmark against ANN-to-SNN conversion and direct SNN training baselines (see Appendix E for implementation details).

Streaming Object Detection. We assess streaming detection by zero-shot transferring models pretrained frame-by-frame on MS COCO (Lin et al., 2014b) directly to the Argoverse-HD benchmark (Li et al., 2020b). COCO provides large-scale object detection pretraining across 80 classes, while Argoverse-HD provides high-frame-rate (30 FPS) egocentric video sequences annotated at 2D bounding-box level across eight road-object overlapping categories (24 validation streams, 15,062 frames). We report frame-wise mAP@50 evaluated sequentially over streaming frames. For conversion baselines, we adapt YOLOv8 (Jocher et al., 2023) for SNN conversion (60.7% AP50 on COCO, ReLU1). Full experimental setups are in Appendix F.

Streaming Automatic Speech Recognition (ASR). We train and evaluate streaming ASR on the LibriSpeech dataset ($\sim$960 hours of speech across 3–15 s clips) (Panayotov et al., 2015). To simulate a real-time streaming environment, audio signals are processed sequentially in small, fixed-size chunks rather than feeding full utterances simultaneously. Candidate transcriptions are served at a low latency every 33 ms to mirror real-time serving pipelines, with final output decisions committed every 1.28 s to ensure prediction stability. Accuracy is measured via Word Error Rate (WER). We base our conversion backbone on an adapted SummaryMixing architecture (Parcollet et al., 2024) (7.62% WER on LibriSpeech); Detailed model and training specifications are provided in Appendix G.

Gesture Recognition on Event Streams. We benchmark streaming event processing using the standard DVS-Gesture dataset (1,342 event recordings across 11 gesture classes captured by a DAVIS240 sensor) (Amir et al., 2017b). In contrast to offline clip-level classification, we evaluate online streaming inference: the model must continuously feed event streams and classify gestures using only the event history received so far. The task involves 12 target classes, including 11 gesture categories and an additional background (NULL) class. Performance is measured by frame-wise classification accuracy. As a conversion backbone, we adapt GestureNet (Massa et al., 2020) (a TCN-based model reaching 91.1% streaming accuracy). Complete setup details are in Appendix H.

Results. Table 1 shows that LIF-Stream generally achieves the best efficiency-accuracy trade-off among SNN techniques: (i) it is substantially more tick-efficient than conversion methods while maintaining peak accuracy, and (ii) it achieves higher peak accuracy with competitive low-tick performance compared to direct training methods. Prior conversion approaches degrade sharply in streaming tasks at low tick budgets ($T < 8$). Conversely, while direct training performs well at

Table 1: Comparison of high-accuracy SNN techniques across ANN-to-SNN conversion and direct training approaches. **Calibration-Free**: requires no data-driven post-training calibration. **Binary Spike**: inter-neuron communication only with a single binary spike per tick; No burst or multi-precision spikes. **Standard neurons**: utilizes basic neuronal dynamics (e.g., IF/LIF) supported by conventional neuromorphic chips. **Bold** and underline denote the best accuracy within the same time-step, across all methods and among conversion methods, respectively.

Method	Calibration Free	Binary Spike	Standard Neurons	Task Accuracy (T is time-steps per frame)						
				$T=2$	$T=4$	$T=8$	$T=16$	$T=32$	$T=64$	Best
Streaming Object Recognition (COCO → ArgoVerse-HD Zero-shot Transfer, Accuracy metric: AP50 mAP@50, higher the better)										
EMS-YOLO (Su et al., 2023)	O	O	O	**20.2**	**22.1**	**22.5**	22.7	23.2	23.6	23.7
SpikeYolo (Luo et al., 2024)	O	✗	✗	-	30.3	-	-	-	-	30.3
ReLUNorm (Rueckauer et al., 2017)	✗	O	O	0.0	0.0	0.0	0.0	0.0	0.9	33.6
Spike-Calib (Li et al., 2021)	✗	O	O	0.0	0.0	0.0	0.1	2.5	10.8	34.6
QCFS (Bu et al., 2023)	✗	O	O	0.0	0.0	1.5	4.2	9.3	25.5	34.1
SlipReLU (Jiang et al., 2023)	✗	O	O	0.0	0.0	0.2	0.8	2.9	12.0	26.0
RMP-SNN (Hao et al., 2023a)	✗	O	O	0.0	0.0	0.0	0.0	0.0	0.0	29.8
DCGS (Huang et al., 2025)	✗	O	✗	0.0	0.0	0.0	0.7	3.8	12.6	37.3
AdaFire (Wang et al., 2025)	✗	✗	✗	0.0	0.0	0.0	0.4	6.4	20.7	35.4
AdaFire+SSC (Wang et al., 2025)	✗	✗	✗	0.0	0.0	0.1	1.4	7.4	17.8	25.0
LIF Neuron + Per-frame Reset	O	O	O	0.0	0.0	0.0	0.0	0.0	25.7	37.4
LIF-Stream (Ours)	O	O	O	12.3	16.8	21.5	**25.2**	**26.5**	**27.8**	**37.4**
Streaming Automatic Speech Recognition (LibriSpeech dev-clean [greedy, no LM], Accuracy metric: WER, lower the better)										
ReLUNorm (Rueckauer et al., 2017)	✗	O	O	100.0	100.0	100.0	100.0	100.0	100.0	72.6
RTS (Deng & Gu, 2021)	✗	O	O	100.0	100.0	100.0	100.0	100.0	100.0	62.9
Spike-Calib (Li et al., 2021)	✗	O	O	100.0	100.0	100.0	100.0	98.1	91.4	65.7
QCFS (Bu et al., 2023)	✗	O	O	114.9	95.9	61.7	30.2	21.1	18.8	18.2
SlipReLU (Jiang et al., 2023)	✗	O	O	122.9	81.5	42.7	23.9	18.1	16.2	14.9
RMP-SNN (Hao et al., 2023a)	✗	O	O	100.0	100.0	100.0	100.0	100.0	94.2	18.3
DCGS (Huang et al., 2025)	✗	O	✗	143.9	108.7	95.9	66.2	27.9	15.1	11.9
AdaFire (Wang et al., 2025)	✗	✗	✗	97.1	94.0	51.0	19.1	12.2	10.5	10.1
AdaFire+SSC (Wang et al., 2025)	✗	✗	✗	99.8	99.2	58.4	19.4	**11.9**	10.5	10.8
LIF Neuron + Per-frame Reset	O	O	O	100.0	100.0	99.9	87.6	12.1	9.9	8.1
LIF-Stream (Ours)	O	O	O	**44.6**	**27.8**	**19.3**	**15.5**	12.0	**9.8**	**8.0**
Event Stream-based Gesture Recognition (DVS-Gesture, Accuracy metric: Multi-class Accuracy over 11+1 classes, higher the better)										
STBP (Wu et al., 2018)	O	O	O	28.2	87.5	**90.2**	**90.6**	**90.5**	90.3	90.6
TET (Deng et al., 2022)	O	O	O	65.1	**88.4**	89.7	89.7	89.5	89.5	89.7
OTTT (Xiao et al., 2022)	O	O	O	19.8	86.3	89.7	90.3	90.3	90.3	90.3
SLTT (Meng et al., 2023)	O	O	O	37.6	86.7	88.5	88.8	88.8	88.9	88.9
ASGL (Wang et al., 2023)	O	O	✗	36.3	86.9	89.6	89.8	89.7	89.7	89.8
SDT (Yao et al., 2023)	O	O	O	-	-	-	89.4	-	-	89.4
ReLUNorm (Rueckauer et al., 2017)	✗	O	O	25.2	25.0	25.0	4.8	7.5	28.4	86.9
RTS (Deng & Gu, 2021)	✗	O	O	12.6	4.8	4.8	4.8	5.9	25.2	86.9
Spike-Calib (adv) (Li et al., 2021)	✗	O	O	25.0	27.5	49.2	78.8	86.2	87.5	88.6
QCFS (Bu et al., 2023)	✗	O	O	59.1	69.5	85.9	88.3	89.6	90.4	90.8
SlipReLU (Jiang et al., 2023)	✗	O	O	56.8	63.7	79.5	87.4	89.3	90.4	90.8
RMP-SNN (Hao et al., 2023a)	✗	O	O	25.0	14.3	4.8	4.8	4.8	18.5	26.5
DCGS (Huang et al., 2025)	✗	O	✗	25.0	25.0	25.0	25.0	27.0	53.3	86.0
AdaFire (Wang et al., 2025)	✗	✗	✗	12.6	25.0	24.8	4.8	9.0	31.1	87.0
AdaFire+SSC (Wang et al., 2025)	✗	✗	✗	12.6	25.0	25.0	4.8	4.8	16.2	85.8
LIF Neuron + Per-frame Reset	O	O	O	13.8	19.8	19.8	20.8	88.2	**90.5**	**90.9**
LIF-Stream (Ours)	O	O	O	**66.4**	80.3	88.5	90.0	90.4	**90.5**	**90.9**

ultra-low ticks, its peak accuracy remains constrained, and it fails to train on complex backbones matching conversion baselines beyond DVS-Gesture (e.g., streaming object detection and ASR).

Importantly, LIF-Stream is natively compatible with standard digital neuromorphic hardware supporting soft reset, as it operates out-of-the-box without post-training model calibration, customized neuronal dynamics, or non-binary spike communication. These structural properties also preserve the full ANN-level peak accuracy ceiling of our method. Prior conversion methods frequently sacrifice peak accuracy to compensate for low-tick latency (via parameter altering (Li et al., 2021) or discontinuous activation functions (Bu et al., 2023; Jiang et al., 2023)). Similarly, direct SNN training suffers from an inherent accuracy gap with ANNs due to discontinuous spiking mechanisms and surrogate gradient approximations, which trade smoothness for exactness.

This tick-efficiency stems from our incremental computing approach. Streaming inference tasks frequently exhibit high inter-frame similarity, which LIF-Stream exploits to reduce the average computational cost required for the SNN to track the reference ANN prediction. Consequently, our framework achieves high accuracy with low frame-wise ticks. This reduction is vital for real-time streaming, where per-frame computing load directly bottlenecks the system throughput, making low tick latency essential for SNNs in streaming workloads.

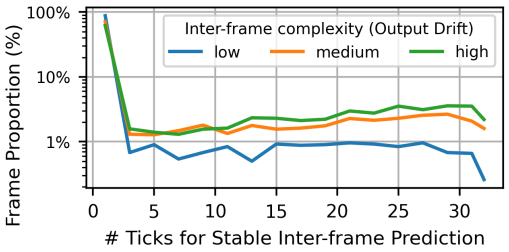


Figure 3: Efficiency via Inter-frame Redundancy.

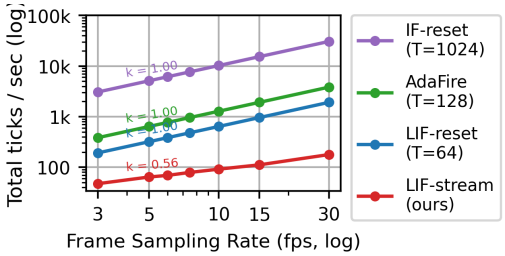


Figure 4: Finer temporal resolution benefits ours.

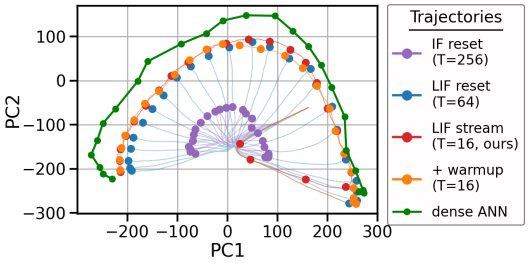


Figure 5: SNNs on egocentric driving perception.

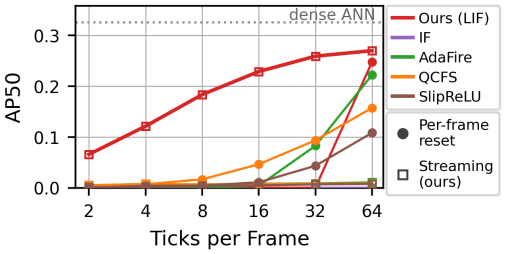


Figure 6: Prior IF-conversions fail to stream.

4.1 In-depth Analysis

We comprehensively analyze LIF-Stream across multiple dimensions. Setup details are in Appendix I.

Inter-frame Redundancy benefits LIF-Stream. We demonstrate the efficiency gains of our incremental computing SNN by measuring the distribution of inter-frame ticks required to reach a stable, confident prediction. Specifically, we conduct a post-hoc evaluation on LIF-Stream using Argoverse dataset and measure, for each frame, the number of ticks needed to fall and remain within an $\epsilon = 0.15$ error tolerance of the peak SNN F1 score. We define frame complexity as the prediction drift of the reference ANN, given by the relative variation $\|y^* - y^*_{\text{prev}}\|/\|y^*_{\text{prev}}\|$. As shown in Figure 3, a logarithmically dominant portion of frames converge in fewer than three ticks due to temporal redundancy. This effect becomes even more pronounced for lower-complexity frames, confirming that SNNs efficiently process consecutive frames by reusing prior prediction states.

Finer temporal resolution benefits LIF-Stream. To further demonstrate computational efficiency, we examine how the tick cost required for accurate predictions scales with temporal resolution. Using the Argoverse dataset, we perform streaming SNN inference over fixed time intervals and aggregate the ticks needed per frame to reach the stable prediction region (within $\epsilon = 0.15$ of the peak SNN F1 score). We then compute the cumulative tick consumption per second across varying temporal resolutions. As shown in Figure 4, reset-based SNN baseline scales linearly ($\mathcal{O}(T^{1.0})$) with temporal resolution because it discards prior state information and recomputes predictions from scratch for every frame. In contrast, LIF-Stream exhibits sub-linear scaling ($\mathcal{O}(T^{0.56})$). Higher temporal resolution increases inter-frame similarity, reducing LIF-Stream's computation per frame.

Qualitative Evaluation. Figure 5 illustrates the streaming inference trajectory to highlight how LIF-Stream efficiently handles consecutive frames unlike reset-based frame-wise approaches. Inference is evaluated on a stream from Argoverse-HD, featuring complex egocentric driving scenes in contrast to the static person-detection scenarios in Figure 2. Reset-based methods reinitialize their hidden states to the origin at every frame, requiring them to cover a significantly longer distance in state space per frame. In contrast, LIF-Stream computes small state updates between the prior state and the new target prediction, effectively exploiting inter-frame prediction deltas. As a result, it converges faster with fewer ticks. Additionally, our initial warm-up mechanism positions the SNN state near the ANN output at the first frame, effectively preventing early mispredictions.

Compatibility with IF-SNN conversions. Our approach is conceptually orthogonal to advanced conversion techniques that calibrate or optimize IF-SNNs, as it is built upon a distinct theoretical foundation. We evaluate our streaming inference framework alongside existing ANN-to-SNN conversion methods to examine how these techniques interact. Specifically, we apply our framework across diverse conversion methods (while maintaining IF neurons) and measure their AP50 accuracy over ticks on Argoverse-HD to determine if incremental computing benefits advanced conversion

techniques. As shown in Figure 6, our streaming SNN inference does not transfer to existing IF-neuron-based conversion methods. Although these techniques are engineered to accelerate early-step inference in static frames, their gains do not propagate across frames, as our streaming framework relies theoretically on LIF neuron dynamics. Consequently, this demonstrates that streaming SNN inference calls for tailored optimizations beyond standard static IF conversion approaches.

5 Related Works

ANN-to-SNN Conversion is an alternative of direct training to build a high-performance SNN that can match the ANN accuracy. Cao et al. (2015) and Diehl et al. (2015) first empirically demonstrated its feasibility. Rueckauer et al. (2016; 2017) established its theoretical basis by finding a steady-state equivalence between discrete IF neuron firing rates and clipped ReLU activations. Because early conversion methods took too many time steps even for a single static input, subsequent works focused on improving the static SNN accuracy in early steps (Diehl et al., 2015; Li et al., 2021; Hao et al., 2023b). Deng & Gu (2021) formulated the layer-wise conversion errors, and Bu et al. (2023) categorizes this error to propose a clip-floor function to replace ReLU1. Huang et al. (2025) propose a new differential multi-threshold neuron that communicates rate delta, and Hwang et al. (2024) converted Transformers using a custom winner-takes-all neurons and one-spike phase coding.

We contribute to the conversion theory in two novel directions: (i) we reveal the principle underlying the transient SNN dynamics which fundamentally enabled the ANN-to-SNN conversion, and (ii) we make the converted SNNs efficiently process the sequence or streaming inputs without any additional training or fine-tuning. We also challenge the conventional dependence on IF neurons, showing why LIF neurons should be primarily used for stream processing. Kugele et al. (2020) proposed an ANN-to-SNN conversion technique for streaming workloads using *streaming rollout*: Similar to pipelining, a signal propagates only a block per frame, whereas conventional sequential rollouts propagates full blocks per frame. This pipelining approach is orthogonal to our incremental computing framework.

SNN Theories. Continuous-time SNNs have been widely studied theoretically in terms of computational power (Maass, 1996a;b) or dynamics (Gerstner et al., 2014; Zhang et al., 2021; Mancoo et al., 2020). However, directly applying these results to discrete-time SNNs causes errors that accumulate over time (Mancoo et al., 2020; Niesen & Hall, 2004). Discontinuous discrete dynamics are more difficult to analyze, and hence relatively few works have studied discrete-time SNNs; through symbolic dynamics (Cessac, 2008), ergodic theory (Cessac, 2011), geometric expressivity (Nguyen et al., 2025), or optimization theory (Oh & Lee, 2024). We provide a new perspective by viewing the transient dynamics of discrete SNNs as continuous attraction toward an instantaneous ANN. This bridges transient SNN dynamics, time-varying attractor dynamics, and the conversion theory.

Sigma-delta modulation (de Jagar, 1952; Steele, 1975; O'Connor & Welling, 2016) conceptually links to our work. In detail, Yoon (2016) interpreted LIF neuron as a variant of sigma-delta modulator, an analog-to-digital converter that communicates only quantized temporal differences between nodes. Following works applied this idea for sequence processing DNNs by computing only quantized activation deltas (O'Connor & Welling, 2016; O'Connor et al., 2017; Zambrano et al., 2019). In contrast, we view SNN computation through the lens of attractor dynamics. The concept of closing the temporal activation gap is a byproduct of controlling SNN dynamics for efficient stream processing.

We further discuss related works on direct SNN training and neural delta networks in Appendix B.

6 Conclusion

We reveal a core principle underlying discrete SNN dynamics: an instantaneous ReLU1-ANN serves as a moving reference target that drives the output trajectory of the discrete SNN. Upon this, we build LIF-Stream, an efficient stream processing framework that controls a discrete SNN to continuously adapting to the reference ANN's output trajectory, thereby incrementally computing the streaming perception. Comprehensive evaluations across three large-scale benchmarks with distinct modalities show that our method achieves the best efficiency-accuracy trade-off. LIF-Stream is deployable out-of-the-box only with a trained ReLU1-ANN, requiring no post-training model calibration, customized neuron dynamics, or non-binary spike formats. Broader discussions, including limitations regarding hardware evaluations and sequence modeling, are detailed in Appendix A.

AI USE STATEMENT

In this work, we used generative AI tools to assist in the writing of proofs, and design or provide feedback on research methodology or experiments. We have not used generative AI tools to:

- help develop theoretical models or conceptual frameworks,
- formulate mathematical claims,
- provide critical ingredients for proving mathematical claims,
- propose or refine hypotheses,
- implement methods,
- assist with translation, or
- interpret results.

and [generate synthetic data sets, clean and reformat dataset, support qualitative and thematic data analysis] are **not applicable** to this work.

Additionally, we used generative AI tools to:

- create or modify scientific figures or images,
- edit a research paper to improve readability, and
- create or edit baseline codes and experimental scripts.

We have reviewed all AI-assisted work by manually verifying the proofs extensively, testing LLM-generated experimental scripts, and proofreading readability-assisted manuscript thoroughly by all authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We have transparently disclosed all experimental setups across both the main paper and the technical appendices. Specifically, within the main text, we describe the setups in the 'empirical validation' paragraphs of Section 2 and Section 3, the core experimental protocols at the beginning of Section 4, and the in-depth analyses setups in Section 4.1. Extensive details regarding each streaming task and its corresponding model configurations are provided in Appendix F (streaming object detection), Appendix G (streaming ASR), and Appendix H (event-stream gesture recognition). Furthermore, the implementation details for baseline methods and ours are thoroughly documented in Appendix E, while setup details for the in-depth analyses are further elaborated in Appendix I. Full proofs for all theoretical results are written in Appendix C, and the complete code is provided via the supplementary materials.

REFERENCES

Filipp Akopyan, Jun Sawada, Andrew Cassidy, Rodrigo Alvarez-Icaza, John Arthur, Paul Merolla, Nabil Imam, Yutaka Nakamura, Pallab Datta, Gi-Joon Nam, et al. Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE transactions on computer-aided design of integrated circuits and systems*, 34(10):1537–1557, 2015.

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 Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7243–7252, 2017a.

Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo Di Nolfo, Tapan Nayak, Alexander Andreopoulos, Guillaume Garreau, Marcela Mendoza, et al. A low power, fully event-based gesture recognition system. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 7388–7397. IEEE, 2017b.

David Attwell and Simon B Laughlin. An energy budget for signaling in the grey matter of the brain. *Journal of Cerebral Blood Flow & Metabolism*, 21(10):1133–1145, 2001.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *Advances in neural information processing systems*, 32, 2019.

Guillaume Bellec, Franz Scherr, Anand Subramoney, Elias Hajek, Darjan Salaj, Robert Legenstein, and Wolfgang Maass. A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature communications*, 11(1):3625, 2020.

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. *arXiv preprint arXiv:2303.04347*, 2023.

Mark Buckler, Philip Bedoukian, Suren Jayasuriya, and Adrian Sampson. Eva^2: Exploiting temporal redundancy in live computer vision. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pp. 533–546. IEEE, 2018.

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.

Lukas Cavigelli, Philippe Degen, and Luca Benini. Cbinfer: Change-based inference for convolutional neural networks on video data. In *Proceedings of the 11th international conference on distributed smart cameras*, pp. 1–8, 2017.

Bruno Cessac. A discrete time neural network model with spiking neurons: rigorous results on the spontaneous dynamics. *Journal of Mathematical Biology*, 56:311–345, 2008.

Bruno Cessac. A discrete time neural network model with spiking neurons: Ii: Dynamics with noise. *Journal of mathematical biology*, 62(6):863–900, 2011.

Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, and James Hays. Argoverse: 3d tracking and forecasting with rich maps. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8748–8757, 2019.

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

F de Jagar. Delta modulation, a method of pcm transmission using a one-unit code. *Philips Research Rept.*, 7:442, 1952.

Shikuang Deng and Shi Gu. Optimal conversion of conventional artificial neural networks to spiking neural networks. *arXiv preprint arXiv:2103.00476*, 2021.

Shikuang Deng, Yuhang Li, Shanghang Zhang, and Shi Gu. Temporal efficient training of spiking neural network via gradient re-weighting. *arXiv preprint arXiv:2202.11946*, 2022.

Peter U Diehl, Daniel Neil, Jonathan Binas, Matthew Cook, Shih-Chii Liu, and Michael Pfeiffer. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *2015 International joint conference on neural networks (IJCNN)*, pp. 1–8. ieee, 2015.

Matthew Dutson, Yin Li, and Mohit Gupta. Event neural networks. In *European Conference on Computer Vision*, pp. 276–293. Springer, 2022.

Biyi Fang, Xiao Zeng, and Mi Zhang. Nestdnn: Resource-aware multi-tenant on-device deep learning for continuous mobile vision. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, pp. 115–127, 2018.

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

Wei Fang, Yanqi Chen, Jianhao Ding, Zhaofei Yu, Timothee Masquelier, Ding Chen, Liwei Huang, Huihui Zhou, Guoqi Li, and Yonghong Tian. Spikingjelly: An open-source machine learning infrastructure platform for spike-based intelligence. *Science Advances*, 9(40):eadi1480, 2023. doi: 10.1126/sciadv.adi1480. URL https://www.science.org/doi/abs/10.1126/sciadv.adi1480.

Karl Friston. The free-energy principle: a unified brain theory? *Nature reviews neuroscience*, 11(2): 127–138, 2010.

Karl Friston and Stefan Kiebel. Predictive coding under the free-energy principle. *Philosophical transactions of the Royal Society B: Biological sciences*, 364(1521):1211, 2009.

Steve B Furber, Francesco Galluppi, Steve Temple, and Luis A Plana. The spinnaker project. *Proceedings of the IEEE*, 102(5):652–665, 2014.

Chang Gao, Daniel Neil, Enea Ceolini, Shih-Chii Liu, and Tobi Delbruck. Deltarnn: A power-efficient recurrent neural network accelerator. In *Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 21–30, 2018.

Zhengyang Geng, Xin-Yu Zhang, Shaojie Bai, Yisen Wang, and Zhouchen Lin. On training implicit models. *Advances in Neural Information Processing Systems*, 34:24247–24260, 2021.

Wulfram Gerstner and Werner M Kistler. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.

Wulfram Gerstner, Werner M Kistler, Richard Naud, and Liam Paninski. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

Hector A Gonzalez, Jiaxin Huang, Florian Kelber, Khaleelulla Khan Nazeer, Tim Langer, Chen Liu, Matthias Lohrmann, Amirhossein Rostami, Mark Schöne, Bernhard Vogginger, et al. Spinnaker2: A large-scale neuromorphic system for event-based and asynchronous machine learning. *arXiv preprint arXiv:2401.04491*, 2024.

Yufei Guo, Yuanpei Chen, Xiaode Liu, Weihang Peng, Yuhan Zhang, Xuhui Huang, and Zhe Ma. Ternary spike: Learning ternary spikes for spiking neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 12244–12252, 2024.

Bing Han and Kaushik Roy. Deep spiking neural network: Energy efficiency through time based coding. In *European Conference on Computer Vision*, pp. 388–404. Springer, 2020.

Jianhui Han, Zhaolin Li, Weimin Zheng, and Youhui Zhang. Hardware implementation of spiking neural networks on fpga. *Tsinghua Science and Technology*, 25(4):479–486, 2020.

Rui Han, Qinglong Zhang, Chi Harold Liu, Guoren Wang, Jian Tang, and Lydia Y Chen. Legodnn: Block-grained scaling of deep neural networks for mobile vision. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, pp. 406–419, 2021.

Zecheng Hao, Tong Bu, Jianhao Ding, Tiejun Huang, and Zhaofei Yu. Reducing ann-snn conversion error through residual membrane potential. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 11–21, 2023a.

Zecheng Hao, Jianhao Ding, Tong Bu, Tiejun Huang, and Zhaofei Yu. Bridging the gap between anns and snns by calibrating offset spikes. *arXiv preprint arXiv:2302.10685*, 2023b.

Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500, 1952.

Zihan Huang, Wei Fang, Tong Bu, Peng Xue, Zecheng Hao, Wenxuan Liu, Yuanhong Tang, Zhaofei Yu, and Tiejun Huang. Differential coding for training-free ann-to-snn conversion. *arXiv preprint arXiv:2503.00301*, 2025.

Loc N Huynh, Youngki Lee, and Rajesh Krishna Balan. Deepmon: Mobile gpu-based deep learning framework for continuous vision applications. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pp. 82–95, 2017.

Sangwoo Hwang, Seunghyun Lee, Dahoon Park, Donghun Lee, and Jaeha Kung. Spikedattention: Training-free and fully spike-driven transformer-to-snn conversion with winner-oriented spike shift for softmax operation. *Advances in Neural Information Processing Systems*, 37:67422–67445, 2024.

Eugene M Izhikevich. Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14 (6):1569–1572, 2003.

Haiyan Jiang, Srinivas Anumasa, Giulia De Masi, Huan Xiong, and Bin Gu. A unified optimization framework of ann-snn conversion: towards optimal mapping from activation values to firing rates. In *International Conference on Machine Learning*, pp. 14945–14974. PMLR, 2023.

Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics yolo. *Zenodo*, 2023.

Saeed Reza Kheradpisheh, Mohammad Ganjtabesh, Simon J Thorpe, and Timothée Masquelier. Stdp-based spiking deep convolutional neural networks for object recognition. *Neural Networks*, 99:56–67, 2018.

Dmitry Krotov and John J Hopfield. Dense associative memory for pattern recognition. *Advances in neural information processing systems*, 29, 2016.

Alexander Kugele, Thomas Pfeil, Michael Pfeiffer, and Elisabetta Chicca. Efficient processing of spatio-temporal data streams with spiking neural networks. *Frontiers in neuroscience*, 14:439, 2020.

Souvik Kundu, Massoud Pedram, and Peter A Beerel. Hire-snn: Harnessing the inherent robustness of energy-efficient deep spiking neural networks by training with crafted input noise. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5209–5218, 2021.

Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer. Training deep spiking neural networks using backpropagation. *Frontiers in neuroscience*, 10:508, 2016.

Mengtian Li, Yu-Xiong Wang, and Deva Ramanan. Towards streaming perception. In *European Conference on Computer Vision (ECCV)*, pp. 473–488, 2020a.

Mengtian Li, Yu-Xiong Wang, and Deva Ramanan. Towards streaming perception. In *European conference on computer vision*, pp. 473–488. Springer, 2020b.

Shengzhao Li, Qin Wang, Jianfei Jiang, Weiguang Sheng, Naifeng Jing, and Zhigang Mao. An efficient cnn accelerator using inter-frame data reuse of videos on fpgas. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 30(11):1587–1600, 2022.

Yuhang Li, Shikuang Deng, Xin Dong, Ruihao Gong, and Shi Gu. A free lunch from ann: Towards efficient, accurate spiking neural networks calibration. In *International conference on machine learning*, pp. 6316–6325. PMLR, 2021.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pp. 740–755, 2014a.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014b.

Xinhao Luo, Man Yao, Yuhong Chou, Bo Xu, and Guoqi Li. Integer-valued training and spike-driven inference spiking neural network for high-performance and energy-efficient object detection. In *European Conference on Computer Vision*, pp. 253–272. Springer, 2024.

Wolfgang Maass. Lower bounds for the computational power of networks of spiking neurons. *Neural Computation*, 8:1–40, 1996a.

Wolfgang Maass. Networks of spiking neurons: The third generation of neural network models. *Electron. Colloquium Comput. Complex.*, TR96, 1996b.

Allan Mancoo, Sander Keemink, and Christian K Machens. Understanding spiking networks through convex optimization. *Advances in Neural Information Processing Systems*, 33:8824–8835, 2020.

Riccardo Massa, Alberto Marchisio, Maurizio Martina, and Muhammad Shafique. An efficient spiking neural network for recognizing gestures with a dvs camera on the loihi neuromorphic processor. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9. IEEE, 2020.

Qingyan Meng, Mingqing Xiao, Shen Yan, Yisen Wang, Zhouchen Lin, and Zhi-Quan Luo. Towards memory-and time-efficient backpropagation for training spiking neural networks. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6143–6153. IEEE, 2023.

Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. Mot16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*, 2016.

Javier R Movellan. Contrastive hebbian learning in the continuous hopfield model. In *Connectionist models*, pp. 10–17. Elsevier, 1991.

Arvind Narayanan, Saurabh Verma, Eman Ramadan, Pariya Babaie, and Zhi-Li Zhang. Deepcache: A deep learning based framework for content caching. In *Proceedings of the 2018 Workshop on Network Meets AI & ML*, pp. 48–53, 2018.

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

Daniel Neil, Jun Haeng Lee, Tobi Delbruck, and Shih-Chii Liu. Delta networks for optimized recurrent network computation. In *International conference on machine learning*, pp. 2584–2593. PMLR, 2017.

Duc Anh Nguyen, Ernesto Araya, Adalbert Fono, and Gitta Kutyniok. Time to spike? understanding the representational power of spiking neural networks in discrete time. *arXiv preprint arXiv:2505.18023*, 2025.

Jitse Niesen and Trinity Hall. *On the global error of discretization methods for ordinary differential equations*. PhD thesis, University of Cambridge, 2004.

Peter O'Connor and Max Welling. Sigma delta quantized networks. *arXiv preprint arXiv:1611.02024*, 2016.

Peter O'Connor, Efstratios Gavves, and Max Welling. Temporally efficient deep learning with spikes. *arXiv preprint arXiv:1706.04159*, 2017.

Hyunseok Oh and Youngki Lee. Sign gradient descent-based neuronal dynamics: Ann-to-snn conversion beyond relu network. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 38562–38598, 2024.

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 *2021 IEEE Workshop on Signal Processing Systems (SiPS)*, pp. 254–259. IEEE, 2021.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: An ASR corpus based on public domain audio books. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5206–5210, 2015.

Titouan Parcollet, Rogier van Dalen, Shucong Zhang, and Sourav Bhattacharya. SummaryMixing: A linear-complexity alternative to self-attention for speech recognition and understanding. In *Interspeech*, 2024.

Mathias Parger, Chen Tang, Christopher D. Twigg, Cem Keskin, Robert Wang, and Markus Steinberger. Deltacnn: End-to-end cnn inference of sparse frame differences in videos. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12487–12496, 2022.

Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Thomas Adler, Lukas Gruber, Markus Holzleitner, Milena Pavlović, Geir Kjetil Sandve, et al. Hopfield networks is all you need. *arXiv preprint arXiv:2008.02217*, 2020.

Rajesh PN Rao and Dana H Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1):79–87, 1999.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

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

Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, and Michael Pfeiffer. Theory and tools for the conversion of analog to spiking convolutional neural networks. *arXiv preprint arXiv:1612.04052*, 2016.

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.

Benjamin Scellier and Yoshua Bengio. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience*, 11:24, 2017.

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.

Shuaijie Shen, Chao Wang, Renzhuo Huang, Yan Zhong, Qinghai Guo, Zhichao Lu, Jianguo Zhang, and Luziwei Leng. Spikingssms: Learning long sequences with sparse and parallel spiking state space models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 20380–20388, 2025.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Matei-Ioan Stan and Oliver Rhodes. Learning long sequences in spiking neural networks. *Scientific Reports*, 14(1):21957, 2024.

Raymond Steele. Delta modulation systems. 1975. URL `https://api.semanticscholar.org/CorpusID:60916149`.

Qiaoyi Su, Yuhong Chou, Yifan Hu, Jianing Li, Shijie Mei, Ziyang Zhang, and Guoqi Li. Deep directly-trained spiking neural networks for object detection. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6532–6542. IEEE, 2023.

Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd international conference on pattern recognition (ICPR)*, pp. 2464–2469. IEEE, 2016.

Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *2019 IEEE/CVF international conference on computer vision (ICCV)*, pp. 9626–9635. Ieee, 2019.

Ziming Wang, Runhao Jiang, Shuang Lian, Rui Yan, and Huajin Tang. Adaptive smoothing gradient learning for spiking neural networks. In *International conference on machine learning*, pp. 35798–35816. PMLR, 2023.

Ziqing Wang, Yuetong Fang, Jiahang Cao, Hongwei Ren, and Renjing Xu. Adaptive calibration: A unified conversion framework of spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 1583–1591, 2025.

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

Mingqing Xiao, Qingyan Meng, Zongpeng Zhang, Yisen Wang, and Zhouchen Lin. Training feedback spiking neural networks by implicit differentiation on the equilibrium state. *Advances in neural information processing systems*, 34:14516–14528, 2021.

Mingqing Xiao, Qingyan Meng, Zongpeng Zhang, D.K. He, and Zhouchen Lin. Online training through time for spiking neural networks. *ArXiv*, abs/2210.04195, 2022. URL https://api.semanticscholar.org/CorpusID:252781051.

Xingrun Xing, Zheng Zhang, Ziyi Ni, Shitao Xiao, Yiming Ju, Siqi Fan, Yequan Wang, Jiajun Zhang, and Guoqi Li. Spikelm: Towards general spike-driven language modeling via elastic bi-spiking mechanisms. *arXiv preprint arXiv:2406.03287*, 2024.

Man Yao, Jiakui Hu, Zhaokun Zhou, Li Yuan, Yonghong Tian, Bo Xu, and Guoqi Li. Spike-driven transformer. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Man Yao, Jiakui Hu, Tianxiang Hu, Yifan Xu, Zhaokun Zhou, Yonghong Tian, Bo Xu, and Guoqi Li. Spike-driven transformer v2: Meta spiking neural network architecture inspiring the design of next-generation neuromorphic chips. In *International Conference on Learning Representations (ICLR)*, 2024.

Bojian Yin, Federico Corradi, and Sander M Bohté. Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks. *Nature Machine Intelligence*, 3(10):905–913, 2021.

Young C Yoon. Lif and simplified srm neurons encode signals into spikes via a form of asynchronous pulse sigma–delta modulation. *IEEE transactions on neural networks and learning systems*, 28(5): 1192–1205, 2016.

Amirreza Yousefzadeh and Manolis Sifalakis. Delta activation layer exploits temporal sparsity for efficient embedded video processing. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pp. 01–10. IEEE, 2022.

Davide Zambrano, Roeland Nusselder, H Steven Scholte, and Sander M Bohté. Sparse computation in adaptive spiking neural networks. *Frontiers in neuroscience*, 12:987, 2019.

Friedemann Zenke and Emre O Neftci. Brain-inspired learning on neuromorphic substrates. *Proceedings of the IEEE*, 109(5):935–950, 2021.

Shao-Qun Zhang, Zhao-Yu Zhang, and Zhi-Hua Zhou. Bifurcation spiking neural network. *The Journal of Machine Learning Research*, 22(1):11459–11479, 2021.

Chenlin Zhou, Liutao Yu, Zhaokun Zhou, Zhengyu Ma, Han Zhang, Huihui Zhou, and Yonghong Tian. Spikingformer: Spike-driven residual learning for transformer-based spiking neural network. *arXiv preprint arXiv:2304.11954*, 2023.

Zhaokun Zhou, Yuesheng Zhu, Chao He, Yaowei Wang, Shuicheng Yan, Yonghong Tian, and Li Yuan. Spikformer: When spiking neural network meets transformer. *arXiv preprint arXiv:2209.15425*, 2022.

Yaoyu Zhu, Zhaofei Yu, Wei Fang, Xiaodong Xie, Tiejun Huang, and Timothée Masquelier. Training spiking neural networks with event-driven backpropagation. *Advances in Neural Information Processing Systems*, 35:30528–30541, 2022.

TECHNICAL APPENDICES

A DISCUSSIONS AND LIMITATIONS

On-device Always-on Inference. A prominent application of this technique is on-device continuous inference. These workloads are typically embodied and event-driven, meaning that sensory inputs are temporally correlated rather than batched. Crucially, surprising events are usually rare, and a proper, composed reaction is often more adequate than immediate precision. Computational resources are scarce on mobile devices; the static amortized cost must remain low while dynamically scaling up when sudden events occur abruptly.

Our technique precisely satisfies these conditions: our average-case cost reduction, combined with the energy efficiency of neuromorphic computing, leads to ultra-low cost in static scenes with high temporal redundancy, while adaptively allocating higher computation only during surprising events.

In contrast, standard ANNs consume significant energy on mobile devices, making true always-on execution infeasible unless they are (i) constantly plugged into power, or (ii) triggered on-demand by humans or lightweight wake-up models. However, the trigger-based approach (ii) either loses immediate always-on responsiveness or inherently sacrifices the accuracy of the main model during non-triggered intervals. By contrast, our technique enables SNNs to maintain permanent responsiveness with ultra-low latency, without sacrificing model accuracy. It is thus an enabling technique for ubiquitous, low-cost, always-on sensing applications, and we demonstrate its practical value by implementing an end-to-end, high-accuracy continuous SNN sensing system on a neuromorphic chip.

Biological Implications. Since SNNs serve as simplified models of biological neural networks, our theory offers valuable insights into how biological brains efficiently sense, perceive, and react continuously throughout their lifespan. Biological brains do not reprocess full-resolution sensory inputs for every inference, nor do they exhibit global state resets. Like SNNs, the brain may persistently retain neural activations and compute only to minimize the error between current spike-coded activations and underlying continuous-valued signals. This mechanism conserves energy in static environments while maintaining constant readiness to respond to unexpected events.

In this perspective, the intrinsic leakage in LIF neurons enables unbounded stream processing without state resets, analogous to how biological forgetting via neuronal leakage clears capacity for new information, thereby facilitating lifelong inference in biological agents. Nevertheless, biological neuronal dynamics are continuous and substantially more complex, and hence extending our theoretical framework to continuous-time and biophysically detailed models, such as the Hodgkin-Huxley (Hodgkin & Huxley, 1952), adaptive LIF (Gerstner et al., 2014), or Izhikevich (Izhikevich, 2003) models, remains an essential direction for future work to validate this hypothesis.

Predictive Coding. Our theoretical framework aligns closely with the principles of predictive coding (Friston & Kiebel, 2009). Predictive coding views the brain as an active *prediction engine*: rather than constructing perception purely bottom-up by processing raw sensory inputs, the brain uses internal generative models to continuously predict incoming signals. Within our formulation, the top-down predictions correspond to the prior SNN output $y(t)$, while the bottom-up prediction error represents the activation gap $y^*(t) - y(t)$ relative to the target ANN output $y^*(t)$. The internal state is subsequently updated through adaptation dynamics that drive $y(t)$ toward $y^*(t)$.In this sense, the neural network computes only the prediction error rather than re-evaluating the full input scene, directly analogous to how our SNN operates exclusively to close the ANN-SNN activation gap. This error-driven scheme ensures extreme bandwidth and energy efficiency by transmitting prediction errors instead of full-resolution sensory data, while enabling rapid response through temporal anticipation. Consequently, our framework provides a formal mechanism demonstrating how predictive coding operates within and benefits SNNs from an ANN-SNN equivalence perspective. An important extension of predictive coding pertains to learning, which we discuss next.

Online Learning for SNNs. Extending our theoretical framework to train the corresponding ANN online over continuous data streams, and thereby train the SNN dynamically, represents a promising research direction. The traditional paradigm of ANN-to-SNN conversion relies on pre-training an ANN offline prior to conversion. However, extending this framework to online learning disentangles ANN-SNN correspondence from a purely static conversion pipeline, establishing this equivalence as a general paradigm for studying SNNs and potentially biological neural networks.

Our theory naturally connects to local, bio-plausible learning methods that assign credit based on per-neuron activation error signals, including predictive coding (Rao & Ballard, 1999; Friston & Kiebel, 2009), equilibrium propagation (Scellier & Bengio, 2017), and contrastive Hebbian learning (Movellan, 1991). While STDP (Kheradpisheh et al., 2018) also measures activation differences over time rather than direct value encodings, its precise theoretical link warrants further investigation.

Prior conversion techniques offer limited implications for online SNN training since they lack an online adaptation perspective; at equilibrium, the corresponding ANN retains few signal components

to exploit beyond static state values. Consequently, training traditionally resorts to standard backpropagation or, for feedback architectures, Deep Equilibrium methods such as implicit differentiation (Bai et al., 2019; Xiao et al., 2021) and phantom gradients (Geng et al., 2021). While DEQ training concepts offer valuable insights, these approaches remain computationally expensive and require global coordination, highlighting the advantages of our adaptive formulation.

Hardware Implementation. Our stream processing SNN framework can be directly deployed on any neuromorphic hardware platform supporting LIF neurons with soft reset. As a cornerstone model in computational neuroscience, the LIF neuron is widely supported across modern neuromorphic platforms, including Intel Loihi (Davies et al., 2018; Orchard et al., 2021), IBM TrueNorth (Akopyan et al., 2015), SpiNNaker (Furber et al., 2014; Gonzalez et al., 2024), and custom FPGA-based implementations (Han et al., 2020). Crucially, our method avoids the need for global state resets, a requirement inherently imposed by prior conversion techniques, allowing each neuron to operate independently without global synchronization beyond a shared clock.

This offers a significant hardware advantage over recent trends aiming for high-accuracy SNNs. Such approaches typically introduce non-standard components alongside basic neuronal dynamics, including synchronized bitwise operations across spike trains (Zhou et al., 2022), customized neuron dynamics (Oh & Lee, 2024), group dynamics (Hwang et al., 2024), or multi-bit spiking mechanisms (Guo et al., 2024; Xing et al., 2024). These additions incur substantial silicon complexity overheads without yielding proportional energy efficiency gains, making custom chip fabrication impractical. Although we evaluate energy efficiency theoretically, following standard, hardware-agnostic evaluation practices in recent theoretical SNN literature (Zhou et al., 2022; 2023; Kundu et al., 2021; Yin et al., 2021; Oh & Lee, 2024), our diverse metrics indicate that these theoretical advantages will readily translate to physical neuromorphic silicon. We leave full-scale hardware deployment as direct future work for efficient mobile applications.

Sequence Processing. Our current framework focuses on converting feedforward architectures to achieve high per-frame accuracy, and thus it does not yet directly apply to sequence models. Nevertheless, extending our theoretical formulation to sequence modeling represents a highly promising research direction for two key reasons: (i) our theory is the first to leverage the transient dynamics of SNNs to enable practically efficient stream processing, and (ii) our framework demonstrates that SNN inference over sequential inputs is inherently stateful.

A motivating insight is that a DEQ can be viewed as a convergent RNN , specifically, an RNN processing a static input over an extended time horizon. Since an SNN at equilibrium corresponds to a DEQ, extending an SNN to time-varying inputs may yield a corresponding RNN with analogous transient dynamics. Alternatively, the transient dynamics and hysteresis behavior of SNNs could be linked to associative memory frameworks, such as Modern Hopfield Networks (Ramsauer et al., 2020; Krotov & Hopfield, 2016), enabling SNNs to execute attention mechanisms or normalized state transitions purely through intrinsic neuronal interactions. Consequently, exploring how SNNs can actively utilize prior state information, rather than merely computing activation deltas, to infer new sequential predictions remains a compelling direction for future work.

B Related Works, cont.

B.1 Direct SNN Training

A common approach for building high-accuracy SNNs is to directly train them with backpropagation-based methods (Lee et al., 2016). The learning signal backpropagates across time-steps, and surrogate gradient (Neftci et al., 2019) relaxes the discontinuous firing rule with smooth approximations. Techniques such as OTTT (Xiao et al., 2022), e-prop (Bellec et al., 2020), mixed-mode differentiations (Zenke & Neftci, 2021), or event-driven BP (Zhu et al., 2022) further improve computational efficiency. BPTT extends to advanced sequence processing architectures like spiking transformers (Zhou et al., 2022; 2023) or state-space SNN models (Stan & Rhodes, 2024; Shen et al., 2025). These prior methods do not compute states incrementally. In contrast, our LIF-Stream reuses prior states to compute only incremental activation updates without requiring specialized training, guaranteeing this property perpetually over unbounded streams. Furthermore, directly trained SNNs

generally exhibit an accuracy gap relative to their ANN counterparts (Fang et al., 2021; Sengupta et al., 2019; Zhou et al., 2022) due to gradient mismatch introduced by surrogate gradient approximations.

B.2 Incremental computing of neural activations

Delta Networks (Neil et al., 2017; Gao et al., 2018; Parger et al., 2022; Yousefzadeh & Sifalakis, 2022) reuses prior activations and incrementally computes only activation differences. However, the efficiency gain only comes when the activation delta is sparsifies through quantization, and thus the ANN should be trained aware of this quantization effect to reduce its impact on stream processing accuracy. Techniques for vision tasks exploit the temporal redundancy in its features by computing its changes from events (Cavigelli et al., 2017; Dutson et al., 2022; Buckler et al., 2018; Li et al., 2022). More coarse-grained approaches adapt neural computations through early exit (Teerapittayanon et al., 2016), test-time block scaling (Fang et al., 2018; Han et al., 2021), or caching (Huynh et al., 2017; Narayanan et al., 2018). Unlike them, our work is a fine-granular neuron-level incremental computing using SNN, and computes to narrow the ANN-SNN activation gap instead of inter-frame ANN activation gap.

C Mathematical Proofs

C.1 Proof of Theorem 2.1

Proof. Let $\alpha := \frac{\tau-1}{\tau}$, so that $\frac{1}{\tau} = 1-\alpha$. Applying the soft-reset rule 3 to the charging equation 1 gives

$$u_{\text{pre}}(t+1) = \alpha \underbrace{\big(u_{\text{pre}}(t) - \theta s(t)\big)}_{=u(t)} + (1-\alpha)\big(u_{rest}\mathbf{1} + I(t+1)\big). \tag{11}$$

The exponential coding scheme defines the output y and filtered input $\tilde{x}$ as exponentially weighted sums of spikes and inputs:

$$y(t) = \sum_{i=1}^{t} \alpha^{t-i}(1-\alpha)\, s(i), \qquad \tilde{x}(t) = \sum_{i=1}^{t} \alpha^{t-i}(1-\alpha)\, x(i). \tag{12}$$

Solving the recurrence equation 11 from $t=0$ forward (using $y(0)=0$ and $s(0)=0$) yields a closed-form relation between u_{pre} and y:

$$u_{\text{pre}}(t) = W_0\, y(t-1) + F\, \tilde{x}(t) + b(t), \tag{13}$$

where $W_0 := W - (\tau-1)\theta\mathbb{I}$ and $b(t) := u_{rest}(1-\alpha^t)\mathbf{1} + \alpha^t\, u_{\text{pre}}(0)$.

To verify equation 13, we expand the accumulated current term using equation 5 and equation 12:

$$\begin{aligned}
\sum_{i=1}^{t+1} \alpha^{t+1-i}(1-\alpha)\, I(i) &= F \sum_{i=1}^{t+1} \alpha^{t+1-i}(1-\alpha)\, x(i) + W \sum_{i=1}^{t+1} \alpha^{t+1-i}(1-\alpha)\, s(i-1) \\
&= F\, \tilde{x}(t+1) + W \sum_{j=0}^{t} \alpha^{t-j}(1-\alpha)\, s(j) \\
&= F\, \tilde{x}(t+1) + W\, y(t), \qquad (\because s(0)=0)
\end{aligned} \tag{14}$$

and substituting this into the inductive solution of equation 11 recovers equation 13.

Finally, substituting equation 13 into the spike-generation rule $y(t) = \alpha\, y(t-1) + (1-\alpha)\, \mathbb{H}(u_{\text{pre}}(t) - \theta\mathbf{1})$ from 4 gives the desired recurrence:

$$y(t) = \alpha\, y(t-1) + (1-\alpha)\, \mathbb{H}\Big(W_0\, y(t-1) + F\, \tilde{x}(t) + b(t) - \theta\mathbf{1}\Big). \tag{15}$$

This is a fixed-point iteration with relaxation factor $(1-\alpha) = \frac{1}{\tau}$. □

C.2 Proof of Theorem 2.2

Proof. From the equation 15, expanding W_0 and dividing by $(\tau-1)\theta$ inside $\mathbb{H}$ (using scale-invariance of $\mathbb{H}$) yields the equivalent form

$$y(t) = y(t-1) - \frac{1}{\tau}\Big(y(t-1) - \mathbb{H}\Big(\frac{F\,\tilde{x}(t) + W\,y(t-1) + b(t) - \theta\cdot\mathbf{1}}{(\tau-1)\theta} - y(t-1)\Big)\Big). \quad (16)$$

We now derive the attractor of this dynamics. For simplicity, we denote the formula inside the step function as

$$D(t) = \frac{1}{(\tau-1)\theta}\big(F\,\tilde{x}(t+1) + Wy(t) + b(t+1) - \theta\cdot\mathbf{1}\big), \quad (17)$$

The **update** $\Delta y(t)$ from the dynamics equation 16 as

$$\Delta y(t) = \mathbb{H}(D(t) - y(t)) - y(t) \quad (18)$$

and the k-th coordinate of the d-dimensional vector as $[\cdot]_k$ for $k \in \{1, \cdots, d\}$. Since the step function $\mathbb{H}$ applies element-wise, we analyze the update coordinate-wise. For any k-th coordinate,

(i) If $\big[D(t)\big]_k < 0$, the k-th coordinate $[y(t)]_k$ updates towards 0.

- $\big[y(t)\big]_k \le \big[D(t)\big]_k < 0$, then $\big[\Delta y(t)\big]_k = 1 - \big[y(t)\big]_k > 0$
- $\big[D(t)\big]_k < \big[y(t)\big]_k < 0$, then $\big[\Delta y(t)\big]_k = 0 - \big[y(t)\big]_k > 0$
- $\big[y(t)\big]_k > 0$, then $\big[\Delta y(t)\big]_k = 0 - \big[y(t)\big]_k < 0$
- $\big[y(t)\big]_k = 0$, then $\big[\Delta y(t)\big]_k = 0 - \big[y(t)\big]_k = 0$

(ii) If $0 \le \big[D(t)\big]_k < 1$, the k-th coordinate $[y(t)]_k$ updates towards $\big[D(t)\big]_k$.

- $\big[y(t)\big]_k < \big[D(t)\big]_k < 1$, then $\big[\Delta y(t)\big]_k = 1 - \big[y(t)\big]_k > 0$
- $1 > \big[y(t)\big]_k = \big[D(t)\big]_k \ge 0$, then $\big[\Delta y(t)\big]_k = 1 - \big[y(t)\big]_k > 0$
- $\big[y(t)\big]_k > \big[D(t)\big]_k \ge 0$, then $\big[\Delta y(t)\big]_k = 0 - \big[y(t)\big]_k < 0$

(iii) If $\big[D(t)\big]_k \ge 1$, the k-th coordinate $[y(t)]_k$ updates towards 1.

- $\big[y(t)\big]_k < 1 \le \big[D(t)\big]_k$, then $\big[\Delta y(t)\big]_k = 1 - \big[y(t)\big]_k > 0$
- $\big[y(t)\big]_k = 1 \le \big[D(t)\big]_k$, then $\big[\Delta y(t)\big]_k = 1 - \big[y(t)\big]_k = 0$
- $1 < \big[y(t)\big]_k \le \big[D(t)\big]_k$, then $\big[\Delta y(t)\big]_k = 1 - \big[y(t)\big]_k < 0$
- $\big[y(t)\big]_k > \big[D(t)\big]_k > 0$, then $\big[\Delta y(t)\big]_k = 0 - \big[y(t)\big]_k < 0$

This applies to all the coordinates. This derivation covers all the possible cases of $y(t)$ and $D(t)$.

So for each k-th coordinate of $y(t)$ and $D(t)$, the following relation holds.

$$[y(t)]_k \to \text{ReLU1}([D(t)]_k) \qquad \text{(Coordinate-wise update direction.)}$$

Geometric Contraction. Fix a coordinate k and let $g(t) := \text{ReLU1}([D(t)]_k)$ be the instantaneous target value. From the relaxed form equation 16, each coordinate evolves as $[y(t)]_k = \alpha\,[y(t-1)]_k + \frac{1}{\tau}\,h$ with $h = \mathbb{H}([D(t)]_k - [y(t-1)]_k) \in \{0, 1\}$ and $\alpha = \frac{\tau-1}{\tau}$.

Suppose $[y(t-1)]_k > g(t) + \frac{1}{\tau}$, i.e., y is above the band. Then $[D(t)]_k - [y(t-1)]_k < 0$, so $h = 0$ and $[y(t)]_k = \alpha\,[y(t-1)]_k$, giving

$$[y(t)]_k - g(t) = \alpha\big([y(t-1)]_k - g(t)\big) - (1-\alpha)\,g(t) \le \alpha\big([y(t-1)]_k - g(t)\big), \quad (19)$$

the gap above contracts by α since $g(t) \ge 0$.

Symmetrically, if $[y(t-1)]_k < g(t) - \frac{1}{\tau}$, i.e., y is below the band, then $[D(t)]_k - [y(t-1)]_k > 0$, so $h = 1$ and $[y(t)]_k = \alpha[y(t-1)]_k + \frac{1}{\tau}$, giving

$$g(t) - [y(t)]_k = \alpha\big(g(t) - [y(t-1)]_k\big) - (1-\alpha)\,(1-g(t)) \le \alpha\big([g(t) - [y(t-1)]_k\big), \quad (20)$$

the gap below contracts by α using $g(t) \leq 1$. Thus, while y lies outside the $\frac{1}{\tau}$-band around $g(t)$, the per-coordinate gap between $y(t)$ and $g(t)$ contracts at rate α.

Once $|[y(t-1)]_k - g(t)| \leq \frac{1}{\tau}$, a single step of magnitude $\frac{1}{\tau}$ cannot leave the $\frac{1}{\tau}$-band since the $y(t)$ step treads across $g(t)$ with a step size $\frac{1}{\tau}$. It may overshoot $g(t)$ but lands within $\frac{1}{\tau}$ on the other side. So y remains trapped, chattering around $g(t)$ without converging to it. This holds for any input trajectory.

The point of coordinate-wise zero-gap $|[y(t)]_k - g(t)| = 0$ between $g(t)$ and $y(t)$ is the point where $[y(t)]_k = \text{ReLU1}([D(t)]_k)$. Since the contraction applies generally for all coordinates, the equilibrium of the contraction is the zero-gap point

$$\left\{ y \in \mathbb{R}^d \mid \forall k,\ \big[y\big]_k = \big[\text{ReLU1}\big(\frac{1}{(\tau-1)\theta}(F\tilde{x}(t+1) + b(t+1) + Wy - \theta \cdot \mathbf{1})\big)\big]_k \right\}, \tag{21}$$

which precisely matches the definition of $\mathcal{E}(t)$:

$$\mathcal{E}(t) = \left\{ y \in \mathbb{R}^d \mid y = \text{ReLU1}\Big(\frac{1}{(\tau-1)\theta}\big(F\tilde{x}(t+1) + Wy + b(t+1) - \theta \cdot \mathbf{1}\big)\Big) \right\}. \tag{22}$$

To sum up, the damped fixed-point iteration with update $\Delta y(t)$ globally contracts the gap with a geometric rate α towards the $\frac{1}{\tau}$ band of the zero-gap equilibrium $\mathcal{E}(t)$ instantaneously defined by $\tilde{x}(t+1)$ and $b(t+1)$. Hence, the LIF-SNN output $y(t)$ is globally attracted towards the instantaneous fixed-point set $\mathcal{E}(t)$, the output of the continuous-valued neural network determined by the input $\tilde{x}(t)$, fed at each time step t. *Global* in the sense that the property holds unconditional to the initial state or prior history. □

C.3 Proof of Corollary 2.3

Proof. We denote $\eta \coloneqq \frac{1}{\tau} = 1 - \alpha$ and $\Phi(y) \coloneqq \text{ReLU1}\big(\frac{1}{(\tau-1)\theta}(F\mathbf{x} + Wy + (u_{rest} - \theta)\mathbf{1})\big)$, so that $\mathcal{E}^* = \{y \mid y = \Phi(y)\}$. Since $y(0) \in [0,1]^d$ and $y(t)$ is an exponential average of spikes started in $[0,1]^d$, $y(t) \in [0,1]^d$ for every t. By equation 16, each step is $y(t) = (1-\eta)\, y(t-1) + \eta\, \mathbb{H}\big(D(t; y(t-1)) - y(t-1)\big)$ with the drive $D(t;y) \coloneqq \frac{1}{(\tau-1)\theta}\big(F\tilde{x}(t) + Wy + b(t) - \theta\mathbf{1}\big)$, whose instantaneous map $\Phi_t(y) \coloneqq \text{ReLU1}(D(t;y))$ has the fixed-point set $\mathcal{E}(t)$. Since ReLU1 is 1-Lipschitz, Φ and every Φ_t are γ-Lipschitz in $\|\cdot\|_\infty$; with $\gamma < 1$ they are contractions on the complete set $[0,1]^d$, and Banach's fixed-point theorem gives $\mathcal{E}^* = \{y^*\}$ and $\mathcal{E}(t) = \{y_t^*\}$.

The geometric rate of approach comes from the one-step contraction of Lemma C.1 below: outside its band, the excess error shrinks by $\rho \coloneqq \max\{\alpha, \gamma\}$ per step. For static input sequences $\tilde{x}(t) \to \mathbf{x}$, the iteration complexity to reach the error tolerance $\epsilon > 0$ is:

$$\rho^t \leq \epsilon \Rightarrow t \geq \frac{\ln(1/\epsilon)}{\ln(1/\rho)}$$

Hence, $t = \mathcal{O}(\log_{1/\rho}(\frac{1}{\epsilon}))$, a linear convergence rate.

Lemma C.1 (One step). *Let Ψ be γ-Lipschitz in $\|\cdot\|_\infty$ with a fixed point $z = \Psi(z) \in [0,1]^d$, and suppose $\sup_y \|\Phi_t(y) - \Psi(y)\|_\infty \leq \varepsilon$. Then*

$$\|y(t) - z\|_\infty \leq \max\Big\{ \underbrace{(1-\eta)\, e}_{\textit{relaxation}},\ \underbrace{\gamma\, e}_{\textit{contraction}} + \underbrace{\eta + \varepsilon}_{\textit{floor}} \Big\}, \qquad e \coloneqq \|y(t-1) - z\|_\infty. \tag{23}$$

Proof. The instantaneous target $g \coloneqq \Phi_t(y(t-1))$ satisfies $\|g - z\|_\infty \leq \|g - \Psi(y(t-1))\|_\infty + \|\Psi(y(t-1)) - \Psi(z)\|_\infty \leq \varepsilon + \gamma e =: r$. Fix a coordinate k and write $v \coloneqq [y(t-1)]_k$, $g_k \coloneqq [g]_k$, $z_k \coloneqq [z]_k \in [0,1]$, and $h \coloneqq \mathbb{H}([D(t; y(t-1))]_k - v)$, so that $[y(t)]_k = (1-\eta)v + \eta h$. By the case analysis in Appendix C.2, $v > g_k$ forces $h = 0$ and $v < g_k$ forces $h = 1$.

- If $v > g_k$: $[y(t)]_k - z_k = (1-\eta)(v - z_k) - \eta\, z_k$, which is at most $(1-\eta)e$ since $z_k \geq 0$, and exceeds $(1-\eta)(z_k - r) - z_k \geq -(\eta + r)$ since $v > g_k \geq z_k - r$ and $z_k \leq 1$.
- If $v < g_k$: symmetrically, $[y(t)]_k - z_k = (1-\eta)(v - z_k) + \eta(1 - z_k)$ lies in $[-(1-\eta)e,\ \eta + r]$.
- If $v = g_k$: $|v - z_k| \leq r$ and the step is at most η, so $|[y(t)]_k - z_k| \leq r + \eta$.

Taking the maximum over k gives equation 23. □

Asymptotic convergence. For a point $y \in \mathbb{R}^d$ and a set $\mathcal{E} \subset \mathbb{R}^d$, denote the point-to-set distance as $\text{dist}_\infty(y, \mathcal{E}) := \inf_{z\in\mathcal{E}} \|y - z\|_\infty$, and the directed Hausdorff semi-distance from $\mathcal{A}$ to $\mathcal{B}$ as $\text{dist}_H(\mathcal{A}, \mathcal{B}) := \sup_{a\in\mathcal{A}} \inf_{b\in\mathcal{B}} \|a - b\|_\infty$. We first derive the triangle equality we will be using in the main proof.

Lemma C.2 (Point-to-set Triangle Inequality)**.** *For any point $y \in \mathbb{R}^d$ and non-empty sets $\mathcal{A}, \mathcal{B} \subset \mathbb{R}^d$, $dist_\infty(y, \mathcal{B}) \le dist_\infty(y, \mathcal{A}) + dist_H(\mathcal{A}, \mathcal{B})$.*

Proof. For any $a \in \mathcal{A}$ and $b \in \mathcal{B}$, the triangle inequality yields $\|y - b\|_\infty \le \|y - a\|_\infty + \|a - b\|_\infty$. Taking $\inf_{b\in\mathcal{B}}$ on both sides gives $\text{dist}_\infty(y, \mathcal{B}) \le \|y - a\|_\infty + \inf_{b\in\mathcal{B}} \|a - b\|_\infty$. Since $\inf_{b\in\mathcal{B}} \|a - b\|_\infty \le \sup_{a'\in\mathcal{A}} \inf_{b\in\mathcal{B}} \|a' - b\|_\infty = \text{dist}_H(\mathcal{A}, \mathcal{B})$ for any $a \in \mathcal{A}$, it follows that $\text{dist}_\infty(y, \mathcal{B}) \le \|y - a\|_\infty + \text{dist}_H(\mathcal{A}, \mathcal{B})$. Finally, taking $\inf_{a\in\mathcal{A}}$ on the right-hand side yields $\text{dist}_\infty(y, \mathcal{B}) \le \text{dist}_\infty(y, \mathcal{A}) + \text{dist}_H(\mathcal{A}, \mathcal{B})$. □

Suppose $\tilde{x}(t) \to \mathbf{x}$. Then $b(t) = u_{rest}(1 - \alpha^t)\mathbf{1} + \alpha^t u_{\text{pre}}(0) \to u_{rest}\mathbf{1}$, since the initial-condition term $\alpha^t u_{\text{pre}}(0)$ decays to zero. The drive then differs from that of Φ only by a vanishing term independent of y,

$$\varepsilon_t := \sup_y \big\|\Phi_t(y) - \Phi(y)\big\|_\infty \le \frac{\|F(\tilde{x}(t) - \mathbf{x}) + b(t) - u_{rest}\mathbf{1}\|_\infty}{(\tau - 1)\theta} \xrightarrow{t\to\infty} 0, \tag{24}$$

and $\mathcal{E}(t) \to \mathcal{E}^*$: from $y_t^* = \Phi_t(y_t^*)$ and $y^* = \Phi(y^*)$, $\|y_t^* - y^*\|_\infty \le \varepsilon_t + \gamma\|y_t^* - y^*\|_\infty$, so $\text{dist}_H(\mathcal{E}(t), \mathcal{E}^*) = \|y_t^* - y^*\|_\infty \le \frac{\varepsilon_t}{1-\gamma}$.

For any desired tolerance $0 < \delta \le \frac{\eta^2}{2}$, we can explicitly construct a finite time $T(\delta, y(0)) = T_1 + T_2$ as follows: By equation 24, there exists a finite $T_1(\delta)$ such that $\text{dist}_H(\mathcal{E}(t), \mathcal{E}^*) < \delta$ for all $t > T_1$.

Let $e_t := \text{dist}_\infty(y(t), \mathcal{E}(t)) = \|y(t) - y_t^*\|_\infty$. Lemma C.1 with $\Psi = \Phi_t$, $z = y_t^*$ and $\varepsilon = 0$, together with $\|y(t-1) - y_t^*\|_\infty \le e_{t-1} + \|y_{t-1}^* - y_t^*\|_\infty < e_{t-1} + 2\delta$ for $t > T_1 + 1$, gives

$$e_t \le \max\big\{(1 - \eta)\, e_{t-1},\ \gamma\, e_{t-1} + \eta\big\} + 2\delta. \tag{25}$$

Let $R_\delta := \frac{\eta+2\delta}{1-\gamma}$, which is at least $\frac{2\delta}{\eta}$ for $\delta \le \frac{\eta^2}{2}$. If $e_{t-1} \le R_\delta$, then $e_t \le R_\delta$; otherwise $e_t - R_\delta \le \rho\,(e_{t-1} - R_\delta)$. Hence there exists a finite number of steps $T_2(\delta, y(0)) \sim \mathcal{O}\big(\log_{1/\rho}(e_{T_1}/\delta)\big)$ required to contract the state error e_t inside the $(R_\delta + \delta)$-neighborhood of $\mathcal{E}(t)$. Triangle inequality (Lemma C.2) then yields $\text{dist}_\infty(y(t), \mathcal{E}^*) \le \text{dist}_\infty(y(t), \mathcal{E}(t)) + \text{dist}_H(\mathcal{E}(t), \mathcal{E}^*) < R_\delta + 2\delta$ for all $t > T_1 + T_2$. Hence, for any arbitrary tolerance $\delta > 0$, there exists a finite time $T(\delta, y(0))$ such that for all $t > T(\delta, y(0))$,

$$\text{dist}_\infty(y(t), \mathcal{E}^*) < \frac{\eta + 2\delta}{1 - \gamma} + 2\delta. \tag{26}$$

Since $\delta > 0$ can be chosen arbitrarily small as $t \to \infty$, this establishes global asymptotic convergence toward the $\frac{1}{\tau(1-\gamma)}$-band of $\mathcal{E}^* = \{y^*\}$:

$$\limsup_{t\to\infty} \|y(t) - y^*\|_\infty \le \frac{\eta}{1 - \gamma} = \frac{1}{\tau(1 - \gamma)}. \tag{27}$$

□

C.4 Theorems and Proofs for IF-SNNs

The Integrate-and-Fire (IF) neuron under rate coding obeys the same three results as the LIF neuron, with one substitution: the relaxation factor η becomes time-dependent. Everything that distinguishes the two neurons for stream processing follows from that single change.

	LIF (exponential coding)	**IF (rate coding)**
relaxation factor	$\eta = \frac{1}{\tau}$ (constant)	$\eta(t) = \frac{1}{t}$ (diminishing)
attraction, any input	Thm. 2.2	Theorem C.4
once the input settles ($\gamma < 1$)	residual $\frac{1}{\tau(1-\gamma)}$ (Cor. 2.3)	residual 0 (Theorem C.5)
tracking a moving target	never decays	decays like $\frac{1}{t}$

Section C.4.1 puts the IF dynamics in fixed-point-iteration form. Section C.4.2 shows that *every* update is directed toward the *time-varying* attractor defined by the input accumulated so far, with no assumption on the input. Section C.4.3 then lets the input settle and shows the residual vanishes, so the convergence is exact. Remark C.7 explains why that exactness is precisely what makes IF-SNNs unsuitable for a perpetual stream.

Setting. An IF neuron with soft reset, threshold θ and unit resistance integrates its input current without leak,

$$u_{\text{pre}}(t) = u(t-1) + I(t), \qquad s(t) = \mathbb{H}\big(u_{\text{pre}}(t) - \theta\mathbf{1}\big), \qquad u(t) = u_{\text{pre}}(t) - \theta\, s(t), \tag{28}$$

with input current $I(t) = Fx(t) + Ws(t-1)$ as in Equation (5) and $s(0) = \mathbf{0}$. Under *rate coding* the output and the filtered input are running arithmetic means,

$$y(t) = \frac{1}{t}\sum_{i=1}^{t} s(i), \qquad \tilde{x}(t) = \frac{1}{t}\sum_{i=1}^{t} x(i), \tag{29}$$

equivalently the streaming recurrence $y(t) = \frac{t-1}{t}y(t-1) + \frac{1}{t}s(t)$ of Equation (4). Two facts are used throughout: $s(i) \in \{0,1\}^d$, so

$$y(t) \in [0,1]^d \quad \text{for every } t, \tag{30}$$

and $\mathbb{H}$ is scale-invariant, $\mathbb{H}(cz) = \mathbb{H}(z)$ for $c > 0$.

C.4.1 Dynamics: IF is a fixed-point iteration with a diminishing step

Lemma C.3. *Let $\tilde{x}(t)$ and $y(t)$ be rate-coded as in equation 29. Then the output trajectory of an IF-SNN is the trajectory of a fixed-point iteration with relaxation factor $\frac{1}{t}$,*

$$y(t) = \frac{t-1}{t}\, y(t-1) + \frac{1}{t}\,\mathbb{H}\bigg(\frac{t-1}{t}\big(W - \theta\mathbb{I}\big)y(t-1) + F\,\tilde{x}(t) + \frac{u_{pre}(0) - \theta\mathbf{1}}{t}\bigg). \tag{31}$$

Proof. The proof is one telescoping sum. Substituting the reset rule of equation 28 into its charging equation gives $u_{\text{pre}}(t) = u_{\text{pre}}(t-1) - \theta\, s(t-1) + I(t)$, hence

$$u_{\text{pre}}(t) = u_{\text{pre}}(0) + \sum_{i=1}^{t} I(i) - \theta\sum_{i=1}^{t-1} s(i). \tag{32}$$

Both sums are rate codes in disguise. With $I(i) = Fx(i) + Ws(i-1)$, $s(0) = \mathbf{0}$ and equation 29,

$$\sum_{i=1}^{t} I(i) = F\sum_{i=1}^{t} x(i) + W\sum_{j=0}^{t-1} s(j) = t\,F\,\tilde{x}(t) + (t-1)\,W\,y(t-1), \qquad \sum_{i=1}^{t-1} s(i) = (t-1)\,y(t-1), \tag{33}$$

so equation 32 becomes the closed form

$$u_{\text{pre}}(t) = (t-1)\big(W - \theta\mathbb{I}\big)\,y(t-1) + t\,F\,\tilde{x}(t) + u_{\text{pre}}(0). \tag{34}$$

Inserting this into $y(t) = \frac{t-1}{t}y(t-1) + \frac{1}{t}\mathbb{H}(u_{\text{pre}}(t) - \theta\mathbf{1})$ and dividing the argument of $\mathbb{H}$ by $t > 0$ gives equation 31. □

Relaxed form and the instantaneous drive. For $t \geq 2$ the factor $\frac{t-1}{t}\theta$ multiplying $y(t-1)$ inside $\mathbb{H}$ in equation 31 is strictly positive, so dividing the argument by it isolates $y(t-1)$ and turns equation 31 into

$$\boxed{y(t) = y(t-1) + \eta(t)\,\Delta y(t), \qquad \Delta y(t) := \mathbb{H}\big(D_{\text{IF}}(t) - y(t-1)\big) - y(t-1), \qquad \eta(t) := \tfrac{1}{t},} \tag{35}$$

driven by the *instantaneous drive*

$$D_{\text{IF}}(t) \;:=\; D_{\text{IF}}\big(t;\, y(t-1)\big), \qquad D_{\text{IF}}(t;y) \;=\; \frac{1}{\theta}\bigg(Wy + \underbrace{\frac{t}{t-1}\,F\,\tilde{x}(t) + \frac{u_{\text{pre}}(0) - \theta\mathbf{1}}{t-1}}_{\text{depends on } t\text{, not on } y}\bigg). \tag{36}$$

Equation 35 is the LIF relaxed form equation 16 with $\frac{1}{\tau}$ replaced by $\eta(t)$; the initial-condition term also decays differently, geometrically as α^t for LIF and as $\frac{1}{t}$ here.

C.4.2 GLOBAL ATTRACTION TOWARD THE INSTANTANEOUS ATTRACTOR

Theorem C.4 (Global attraction, IF). *At each time step $t \geq 2$, the IF-SNN output $y(t)$ is globally attracted with a step size $\eta(t) = \frac{1}{t}$ toward the instantaneous fixed-point set of the continuous-valued neural network,*

$$\mathcal{E}_{IF}(t) \coloneqq \Big\{ y \in [0,1]^d \;\Big|\; y = \mathrm{ReLU1}\big(D_{IF}(t; y)\big) \Big\}. \tag{37}$$

The set $\mathcal{E}_{IF}(t)$ is always nonempty because its ReLU1 map is a continuous self-map on the compact convex domain $[0,1]^d$, guaranteeing the existence of a fixed point by Brouwer's fixed-point theorem.

Proof. We follow the proof of Theorem 2.2 (Appendix C.2), with the constant step size $\frac{1}{\tau}$ replaced by $\eta(t)$. From the relaxed form equation 35, the update is

$$\Delta y(t) = \mathbb{H}\big(D_{\text{IF}}(t) - y(t-1)\big) - y(t-1), \tag{38}$$

and we denote the k-th coordinate of a d-dimensional vector as $[\cdot]_k$ for $k \in \{1, \cdots, d\}$. Since $\mathbb{H}$ applies element-wise, we analyze the update coordinate-wise. Unlike the LIF case, $[y(t-1)]_k \in [0,1]$ always holds by equation 30, so fewer cases arise. For any k-th coordinate,

(i) If $\big[D_{\text{IF}}(t)\big]_k < 0$, the k-th coordinate $[y(t-1)]_k$ updates towards 0.

- $\big[y(t-1)\big]_k > 0$, then $\big[\Delta y(t)\big]_k = 0 - \big[y(t-1)\big]_k < 0$
- $\big[y(t-1)\big]_k = 0$, then $\big[\Delta y(t)\big]_k = 0 - \big[y(t-1)\big]_k = 0$

(ii) If $0 \leq \big[D_{\text{IF}}(t)\big]_k \leq 1$, the k-th coordinate $[y(t-1)]_k$ updates towards $\big[D_{\text{IF}}(t)\big]_k$.

- $\big[y(t-1)\big]_k < \big[D_{\text{IF}}(t)\big]_k$, then $\big[\Delta y(t)\big]_k = 1 - \big[y(t-1)\big]_k > 0$
- $\big[y(t-1)\big]_k = \big[D_{\text{IF}}(t)\big]_k$, then $\big[\Delta y(t)\big]_k = 1 - \big[y(t-1)\big]_k \geq 0$
- $\big[y(t-1)\big]_k > \big[D_{\text{IF}}(t)\big]_k$, then $\big[\Delta y(t)\big]_k = 0 - \big[y(t-1)\big]_k < 0$

(iii) If $\big[D_{\text{IF}}(t)\big]_k > 1$, the k-th coordinate $[y(t-1)]_k$ updates towards 1.

- $\big[y(t-1)\big]_k < 1$, then $\big[\Delta y(t)\big]_k = 1 - \big[y(t-1)\big]_k > 0$
- $\big[y(t-1)\big]_k = 1$, then $\big[\Delta y(t)\big]_k = 1 - \big[y(t-1)\big]_k = 0$

This derivation covers all the possible cases of $y(t-1)$ and $D_{\text{IF}}(t)$, and applies to all the coordinates. So for each k-th coordinate, the following relation holds.

$$[y(t-1)]_k \to \mathrm{ReLU1}([D_{\text{IF}}(t)]_k) \qquad \text{(Coordinate-wise update direction.)}$$

Moreover, $\mathbb{H}(\cdot) \in \{0,1\}^d$ and $y(t-1) \in [0,1]^d$ give $\|\Delta y(t)\|_\infty \leq 1$, so every update moves the output by at most one step, $\|y(t) - y(t-1)\|_\infty \leq \eta(t)$.

Contraction. Fix a coordinate k and let $g(t) \coloneqq \mathrm{ReLU1}([D_{\text{IF}}(t)]_k)$ be the instantaneous target value. From equation 35, each coordinate evolves as $[y(t)]_k = (1 - \eta(t))\,[y(t-1)]_k + \eta(t)\,h$ with $h = \mathbb{H}([D_{\text{IF}}(t)]_k - [y(t-1)]_k) \in \{0,1\}$.

Suppose $[y(t-1)]_k > g(t) + \eta(t)$, i.e., y is above the band. Then $[D_{\text{IF}}(t)]_k - [y(t-1)]_k < 0$, so $h = 0$ and $[y(t)]_k = (1 - \eta(t))\,[y(t-1)]_k$, giving

$$[y(t)]_k - g(t) = (1-\eta(t))\big([y(t-1)]_k - g(t)\big) - \eta(t)\,g(t) \leq (1-\eta(t))\big([y(t-1)]_k - g(t)\big), \tag{39}$$

the gap above contracts by $1 - \eta(t)$ since $g(t) \geq 0$.

Symmetrically, if $[y(t-1)]_k < g(t) - \eta(t)$, i.e., y is below the band, then $[D_{\text{IF}}(t)]_k - [y(t-1)]_k > 0$, so $h = 1$ and $[y(t)]_k = (1 - \eta(t))\,[y(t-1)]_k + \eta(t)$, giving

$$g(t) - [y(t)]_k = (1-\eta(t))\big(g(t) - [y(t-1)]_k\big) - \eta(t)\,(1 - g(t)) \leq (1-\eta(t))\big(g(t) - [y(t-1)]_k\big), \tag{40}$$

the gap below contracts by $1 - \eta(t)$ using $g(t) \leq 1$. Thus, while y lies outside the $\eta(t)$-band around $g(t)$, the per-coordinate gap contracts at rate $1 - \eta(t) = \frac{t-1}{t}$.

Once $|[y(t-1)]_k - g(t)| \leq \eta(t)$, a single step of magnitude at most $\eta(t)$ cannot leave the $\eta(t)$-band. It may overshoot $g(t)$ but lands within $\eta(t)$ on the other side. So y remains trapped, chattering around $g(t)$. In contrast to LIF, the band itself shrinks as $\eta(t) = \frac{1}{t} \to 0$. This holds for any input trajectory.

Unlike the geometric rate α of LIF, the contraction factors now telescope as $\prod_{i=t_0+1}^{t}\left(1-\frac{1}{i}\right)=\frac{t_0}{t}$. Combining the two regimes by induction on t, the per-coordinate gap from any time t_0 onward satisfies

$$\left|[y(t)]_k - g(t)\right| \;\le\; \frac{\max\left\{t_0\left|[y(t_0)]_k - g(t_0)\right|,\,1\right\}}{t} \;=\; \mathcal{O}\Big(\frac{1}{t}\Big). \tag{41}$$

The point of coordinate-wise zero-gap $|[y]_k - g(t)| = 0$ is the point where $[y]_k = \mathrm{ReLU1}([D_{\mathrm{IF}}(t;y)]_k)$. Since the contraction applies to all coordinates, the equilibrium of the contraction is the zero-gap point

$$\left\{ y\in[0,1]^d \;\middle|\; \forall k,\ \big[y\big]_k = \Big[\mathrm{ReLU1}\Big(\frac{1}{\theta}\Big(Wy + \frac{t}{t-1}F\tilde{x}(t) + \frac{u_{\mathrm{pre}}(0)-\theta\mathbf{1}}{t-1}\Big)\Big)\Big]_k \right\}, \tag{42}$$

which precisely matches the definition of $\mathcal{E}_{\mathrm{IF}}(t)$ in equation 37. Finally, the map $y \mapsto \mathrm{ReLU1}(D_{\mathrm{IF}}(t;y))$ is continuous and maps the compact convex set $[0,1]^d$ into itself, so $\mathcal{E}_{\mathrm{IF}}(t)$ is nonempty by Brouwer's fixed-point theorem.

To sum up, the damped fixed-point iteration with update $\Delta y(t)$ contracts the gap with rate $1-\eta(t)$ towards the $\eta(t)$-band of the zero-gap equilibrium $\mathcal{E}_{\mathrm{IF}}(t)$ instantaneously defined by $\tilde{x}(t)$. Hence, the IF-SNN output $y(t)$ is globally attracted towards the instantaneous fixed-point set $\mathcal{E}_{\mathrm{IF}}(t)$. *Global* in the sense that the property holds unconditional to the initial state, prior history, or whether the input $\tilde{x}(t)$ ever settles. □

C.4.3 Exact convergence to the static equilibrium

Now let the input settle. For a fixed limit input $\mathbf{x}$ define the static map and its equilibrium set, which is the set displayed in the main text,

$$\Phi(y) := \mathrm{ReLU1}\Big(\tfrac{1}{\theta}\big(Wy + F\mathbf{x}\big)\Big), \qquad \mathcal{E}_{\mathrm{IF}}(\mathbf{x}) := \big\{y\in[0,1]^d \mid y=\Phi(y)\big\}. \tag{43}$$

Theorem C.5 (Exact convergence, IF)**.** *Let $\gamma := \|W\|_\infty/\theta < 1$. If $\tilde{x}(t)\to\mathbf{x}$ under rate coding, then $\mathcal{E}_{IF}(\mathbf{x})$ is a singleton $\{y^*\}$ and the IF-SNN output converges to it exactly, $\lim_{t\to\infty}\|y(t)-y^*\|_\infty = 0$.*

Proof. We run the argument of Corollary 2.3 (Appendix C.3) with the constant step $\frac{1}{\tau}$ replaced by $\eta_t := \eta(t) = \frac{1}{t}$.

Uniqueness. Φ is γ-Lipschitz in $\|\cdot\|_\infty$, a contraction on $[0,1]^d$ for $\gamma<1$, so Banach's theorem gives $\mathcal{E}_{\mathrm{IF}}(\mathbf{x}) = \{y^*\}$.

The moving target. Comparing equation 36 with equation 43, the two drives share the y-dependent term Wy/θ, so their gap does not depend on y:

$$\varepsilon_t := \sup_{y\in[0,1]^d}\Big\|D_{\mathrm{IF}}(t;y) - \tfrac{1}{\theta}\big(Wy+F\mathbf{x}\big)\Big\|_\infty = \frac{1}{\theta}\Big\|\tfrac{t}{t-1}F\tilde{x}(t) - F\mathbf{x} + \tfrac{u_{\mathrm{pre}}(0)-\theta\mathbf{1}}{t-1}\Big\|_\infty \;\xrightarrow{t\to\infty}\; 0, \tag{44}$$

because $\tilde{x}(t)\to\mathbf{x}$, $\frac{t}{t-1}\to 1$, and the initial-condition term is $\mathcal{O}(\frac{1}{t})$. Since ReLU1 is 1-Lipschitz, the same bound holds after ReLU1.

One step. Let $a_t := \|y(t)-y^*\|_\infty$. The proof of Lemma C.1 uses only $y(t-1), z\in[0,1]^d$, the direction rule of the coordinate-wise case analysis (here Theorem C.4), and the relaxed form of the update (here equation 35). It therefore carries over verbatim to IF with the step η_t, and applying it with $\Psi=\Phi$, $z=y^*$ and $\varepsilon=\varepsilon_t$ gives, for $t\ge 2$,

$$a_t \le \max\big\{(1-\eta_t)\,a_{t-1},\;\; \gamma\,a_{t-1}+\eta_t+\varepsilon_t\big\}. \tag{45}$$

The floor vanishes. Fix $\epsilon>0$ and choose T_ϵ such that, for all $t>T_\epsilon$, $\eta_t+\varepsilon_t\le(1-\gamma)\,\epsilon$ and $1-\eta_t\ge\gamma$, which is possible since $\eta_t\to 0$ and $\varepsilon_t\to 0$. Let $d_t := \max\{a_t-\epsilon, 0\}$. As in the proof of Corollary 2.3, equation 45 gives $d_t\le\max\{\gamma,1-\eta_t\}\,d_{t-1} = (1-\eta_t)\,d_{t-1}$ for $t>T_\epsilon$, and the product telescopes,

$$d_t \;\le\; d_{T_\epsilon}\prod_{i=T_\epsilon+1}^{t}\Big(1-\tfrac{1}{i}\Big) \;=\; d_{T_\epsilon}\,\frac{T_\epsilon}{t} \;\xrightarrow{t\to\infty}\; 0. \tag{46}$$

The product vanishes because $\sum_i \eta_i = \infty$: the step size is diminishing but *non-summable*, which is precisely the Robbins–Monro condition (Robbins & Monro, 1951). Hence $\limsup_t a_t \leq \epsilon$, and as $\epsilon > 0$ was arbitrary, $a_t \to 0$, at the sublinear rate $\mathcal{O}(\frac{1}{t})$ once $t > T_\epsilon$. □

Remark C.6 (One inequality, two step sizes). The step size enters equation 23 and equation 45 only through the floor, so the ultimate residual is $\frac{\lim_t \eta(t)}{1-\gamma}$. For LIF, $\eta \equiv \frac{1}{\tau}$ gives the residual $\frac{1}{\tau(1-\gamma)}$ of Corollary 2.3 at the geometric rate $\max\{\alpha, \gamma\}$; for IF, $\eta(t) = \frac{1}{t} \to 0$ collapses the residual to zero, at the price of an $\mathcal{O}(\frac{1}{t})$ rate. *The exactness of IF conversion and the residual jitter of LIF conversion are one inequality evaluated at two step sizes*, not two separate phenomena.

Remark C.7 (Why exact convergence is a liability on a stream). Theorem C.5 is the dynamical explanation of the steady-state correspondence between IF neurons and clipped ReLU on which ANN-to-SNN conversion has long relied (Rueckauer et al., 2017; Han & Roy, 2020). It also explains why IF-SNNs do not stream. Exactness rests on two hypotheses: the input settles, and the step size vanishes, and the second is a property of the *neuron*, not of the input. By Theorem C.4 the per-tick displacement is at most $\eta(t) = \frac{1}{t}$, so after t ticks of a stream a new frame that moves the reference output by δ needs at least $\delta\, t$ further ticks to be followed: the cost of tracking a change of fixed size grows *linearly with elapsed stream time*, and under a fixed per-frame budget the attainable tracking error grows without bound. This is why conversion pipelines built on IF neurons must reset globally every frame, discarding exactly the state that makes incremental computation cheap. The LIF constant step size trades the exact limit for a permanent $\frac{1}{\tau}$ residual and buys, in exchange, a tracking ability that does not decay with time: the property the streaming framework of Algorithm 1 is built on.

D ALGORITHMS

Algorithm 1 LIF-Stream

Input: LIF-SNN $\mathcal{S}$; Stream $\{x_i\}$; Ticks $\{T_i\}$; Leak τ; Init $y(0), z(0), u_n(0)$
Output: Output stream $\{y_i\}$

Initialize LIF neurons $n \in \mathcal{S}$ with $u_n(0)$
$y \leftarrow y(0)$, $z \leftarrow z(0)$
for each frame x_i in stream **do**
  for $t = 1$ **to** T_i **do**
    $x(t), z \leftarrow \text{StreamEncode}(x_i, z, \tau)$ {Algorithms 2–7}
    $y \leftarrow \frac{\tau-1}{\tau} y + \frac{1}{\tau}\mathcal{S}(x(t))$ {decode}
  end for
  $y_i \leftarrow y$
end for

Stream encoders. The SNN sees its input only through the exponentially coded input $\tilde{x}(t)$ (Theorem 2.2), so any encoder is valid as long as $\tilde{x}$ tracks the frame x. With z the encoder's replica of $\tilde{x}$,

$$x - z \;\leftarrow\; \tfrac{\tau-1}{\tau}\,(x - z) \;+\; \tfrac{1}{\tau}\,\big(x - x(t)\big), \tag{47}$$

i.e., each encoder is a feedback controller on this error, and must emit x on average once $z = x$. The float encoders (Algorithms 2–4) are $x(t) = x + \kappa(x - z)$, contracting the error by $\frac{\tau-1-\kappa}{\tau}$ per tick. The spike encoders (Algorithms 5–7) quantize the same controller: sigma-delta is the LIF dynamics equation 16 with drive x, and population lookahead splits the lookahead current over $K \geq \tau$ neurons with graded thresholds, each firing at most one binary spike per tick, keeping $|x - z| \leq \frac{1}{2\tau}$ for $x \in [0, 1]^d$.

Algorithm 2 Sample-and-hold (float)

Input: Frame x; State z; Leak τ
Output: Current $x(t)$; State z

$x(t) \leftarrow x$ {Stateless encoding}
$z \leftarrow \frac{\tau-1}{\tau} z + \frac{1}{\tau} x(t)$ {$x - z$ decays by $\frac{\tau-1}{\tau}$}
return $x(t),\ z$

Algorithm 3 Proportional residual (float)

Input: Frame x; State z; Leak τ; Gain $\kappa \in [0, \tau - 1]$
Output: Current $x(t)$; State z

$x(t) \leftarrow x + \kappa\,(x - z)$ {Residual feedback}
$z \leftarrow \frac{\tau-1}{\tau} z + \frac{1}{\tau} x(t)$ {$x - z$ decays by $\frac{\tau-1-\kappa}{\tau}$}
return $x(t),\ z$

Algorithm 4 Lookahead (float)

Input: Frame x; State z; Leak τ
Output: Current $x(t)$; State z

$z \leftarrow \frac{\tau-1}{\tau} z$ {Lookahead}
$x(t) \leftarrow \tau\,(x - z)$ {Scaled residual}
$z \leftarrow z + \frac{1}{\tau} x(t)$ {$z = x$ at every tick}
return $x(t),\ z$

Algorithm 5 Sigma-delta (spike)

Input: Frame x; State z; Leak τ
Output: Spikes $x(t)$; State z

$x(t) \leftarrow \mathbb{H}(x - z)$ {Fire if under-coded}
$z \leftarrow \frac{\tau-1}{\tau} z + \frac{1}{\tau} x(t)$ {$|x - z| \lesssim \frac{1}{\tau}$}
return $x(t),\ z$

Algorithm 6 Stochastic sigma-delta (spike)

Input: Frame x; State z; Leak τ; Dither σ
Output: Spikes $x(t)$; State z

$\xi \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbb{I})$
$x(t) \leftarrow \mathbb{H}(x - z + \xi)$ {Dithered firing}
$z \leftarrow \frac{\tau-1}{\tau} z + \frac{1}{\tau} x(t)$
return $x(t),\ z$

Algorithm 7 Population lookahead (spike)

Input: Frame x; State z; Leak τ; Population K
Output: Spikes $\{s_j^{\pm}\}_{j=1}^{K}$; State z

$z \leftarrow \frac{\tau-1}{\tau} z$ {Lookahead}
$r \leftarrow \tau\,(x - z)$ {Float lookahead current}
for $j = 1$ **to** K **do**
  $s_j^{\pm} \leftarrow \mathbb{H}(\pm r - j + \frac{1}{2})$ {Thermometer code}
end for
$x(t) \leftarrow \sum_{j=1}^{K} (s_j^{+} - s_j^{-})$ {$=$ round(r)}
$z \leftarrow z + \frac{1}{\tau} x(t)$ {$|x - z| \leq \frac{1}{2\tau}$}
return $\{s_j^{\pm}\},\ z$

Algorithm 8 ANN → LIF-SNN conversion

Input: ANN g with ReLU1 activations; leak $\tau > 1$; threshold $\theta = 1$
Output: LIF-SNN $\mathcal{S}$, codec initializer $y(0)$, initial $u_n(0)$ for all neurons $n \in \mathcal{S}$

$a_\tau(x) := (\tau - 1)x + 1$
// Main conversion
$\mathcal{S} \leftarrow \textsc{Copy}(g)$
for all DEQ layers $l \in L \subseteq \mathcal{S}$ **do**
  $f \leftarrow$ forward function of l
  for all ReLU1 activations $\sigma \in f$ **do**
    $n \leftarrow \text{LIF}(\tau, \theta)$
    $\sigma \leftarrow n \circ a_\tau$
  end for
  $\text{l} \leftarrow \textsc{Recurrent-SNN}(f)$
end for
for all ReLU1 activations $\sigma \in \mathcal{S} - L$ **do**
  $n \leftarrow \text{LIF}(\tau, \theta)$
  $\sigma \leftarrow n \circ a_\tau$
end for
// Bias correction: pre-charge $u(0)$ with it
Disable firing of every neuron $n \in \mathcal{S}$
Temporarily hook each neuron n to store its influx current b_n
$y(0) \leftarrow \mathcal{S}(\mathbf{0})$
for all neurons $n \in \mathcal{S}$ **do**
  $u_n(0) \leftarrow b_n$
end for
Re-enable firing of neurons
return $\mathcal{S}$

D.1 Comparison of Stream Encoders

We compare the encoders of Algorithms 2–7 at two levels, with $\tau = 8$, $\kappa = 3.5$, $\sigma = 0.4$, and $K = 8$ (Figure 7). *Input side* (a–c): by equation 47 the SNN sees the input only through $\tilde{x}$, so encoding quality is $|x - \tilde{x}|$ at the tick a frame is read out, measured on $d = 512$ coordinates over 64 frames of a random walk in $[0, 1]$ (per-frame step $\mathcal{N}(0, 0.1^2)$, reflected at the boundaries). *Stream accuracy* (d): the converted YOLOv8-style detector of Appendix F streams the first Argoverse-HD validation sequence (901 frames) without reset, and we report AP50 over all frames. Every encoder runs on raw pixels in $[0, 1]$ and feeds the network the normalized current $(x(t) - \mu)/\sigma_{\text{img}}$, which for spike encoders amounts to binary spikes on pixel channels through the normalization folded into the first layer; sample-and-hold is then identical to the streaming arm of the main experiments.

- **Float encoders** follow the contraction $\left(\frac{\tau-1-\kappa}{\tau}\right)^T$ of the gain family, reaching an input error of 2.7×10^{-2} (sample-and-hold), 1×10^{-4} (proportional), and exactly 0 (lookahead) at $T = 8$. On the stream, the faster input translates into faster tracking at small budgets: at $T = 4$, lookahead and proportional reach AP50 0.061 and 0.060 against 0.049 for sample-and-hold, and all three converge to ≈ 0.104 by $T = 32$.
- **Spike encoders** settle at an input error floor set by quantization rather than by T: 0.039 for sigma-delta and 0.033 for population lookahead, which stays below $\frac{1}{2\tau}$ at every tick at the cost of extra spikes only on frame changes (0.73 spikes per coordinate per tick at $T = 1$, 0.52 at $T = 32$, against 0.51 for sigma-delta). On the stream, population lookahead tracks its float counterpart within 0.004 AP50 at every budget, while sigma-delta follows sample-and-hold at small T and levels off lower (0.097 at $T = 32$), the residue of its quantization floor.
- **Stochastic dithering** is the one encoder the detector does not tolerate: its input noise leaves AP50 at 0.036 even at $T = 32$.

This sequence is harder than average: the dense ANN reaches AP50 0.140 on it against 0.296 on the full validation set, so the numbers above are meaningful relative to one another and to that reference rather than in absolute terms.

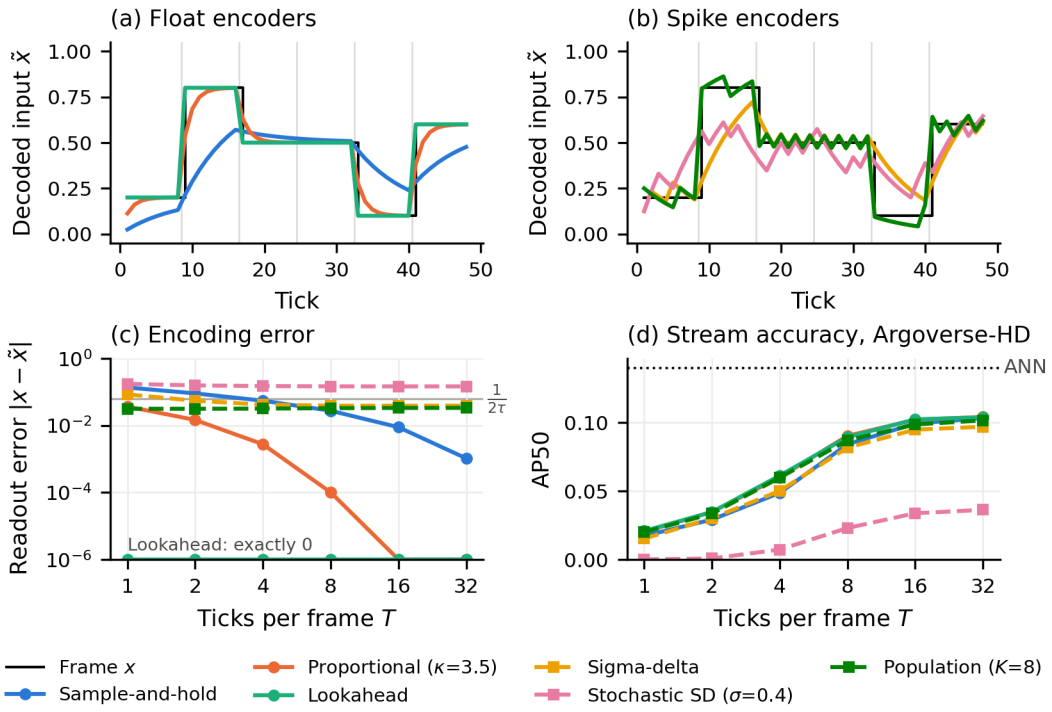


Figure 7: Stream encoders ($\tau = 8$). (a,b) The decoded input $\tilde{x}$ of one coordinate on a staircase of six frames with $T = 8$ ticks each (grey lines mark frame boundaries). (c) Input readout error $|x - \tilde{x}|$ at the last tick of each frame against the tick budget, on a correlated random stream; values are clipped at 10^{-6}. (d) AP50 of the converted detector streaming one Argoverse-HD sequence (901 frames) against the tick budget; the dotted line is the dense ANN.

E Baseline Methods and Implementation Details

This appendix describes how LIF-Stream is implemented (Section E.1), how the baselines are implemented (Section E.2–Section E.3), and the protocol that is shared by every row of Table 1 (Section E.4). All methods are implemented in a single PyTorch codebase on top of *spikingjelly* (Fang et al., 2023), so that every arm of a comparison shares the same data pipeline, the same decoder, and the same metric code. Each baseline is implemented directly from the equations of its own paper; where a published method offers several settings we use the one its authors report as best.

E.1 LIF-Stream

Convertible ANN. Our framework requires only that the reference ANN be a composition of affine operators and the bounded activation $\mathrm{ReLU1}(x)$. Concretely, every architecture we use is built from convolutions, linear layers, batch normalization, average pooling, and ReLU1. Operators that are not affine (e.g. max-pooling, layer normalization, softmax attention) are replaced by affine equivalents before training, so no approximation is introduced at conversion time. Task-specific post-processing that contains no learnable parameters – non-maximum suppression for detection, CTC collapse for speech, arg-max for classification – is executed outside the converted graph, exactly as in the ANN.

Conversion. Conversion follows Algorithm 8. The model is traced into a graph, each Conv–BatchNorm pair is folded into a single affine operator, and each ReLU1 site is replaced by the composition $n \circ a_\tau$ of an input rescaling $a_\tau(x) = (\tau - 1)x + 1$ and a soft-reset LIF neuron with $\theta = 1$, $u_{\text{rest}} = 0$, $R = 1$. Because ReLU1 is bounded by construction, the firing threshold is fixed at 1 and *no activation statistics, calibration set, or per-layer search is required*; this is the only structural difference from the conversion baselines in Section E.2, all of which fit at least one parameter per layer from data. The single data-independent step is the membrane pre-charge of Algorithm 8: firing

is disabled, the zero input is propagated once, and each neuron's resting influx is stored as its initial membrane potential $u_n(0)$ together with the output codec state $y(0)$. This is a one-off correction of the constant term $b(t)$ in Equation (13); it is computed once at conversion time and never re-applied.

Streaming inference. Streaming inference follows Algorithm 1. The membrane potentials of all neurons and the input/output codec states are initialized once at the beginning of a stream and then persist across frames; there is no global reset, no per-frame re-initialization, and no cross-frame communication other than the neuronal state itself. Each frame k is allotted T ticks, and the prediction for frame k is read from the output codec at the last tick of that frame. Inputs are encoded with the sample-and-hold scheme of Algorithm 2: at each tick the encoder emits the current frame unchanged, and the change between frames reaches the SNN through the exponentially coded input $\tilde{x}(t)$, which follows the new frame at the leak rate $\frac{\tau-1}{\tau}$.

Hyperparameters. The framework has two hyperparameters: the leak τ and the per-frame tick budget T. We use $\tau = 8$ throughout (except the Best column), and sweep $T \in \{2, 4, 8, 16, 32, 64\}$, which is the axis reported in Table 1. For the **Best** column, we use $\tau = 16$ and measure the accuracy in $T > 128$, just like all the other baseline methods. An optional *warm-up* grants the first frame of a stream $4T$ extra ticks so that the very first prediction is not read from a cold membrane; its cost is amortized over the whole stream and it is disabled unless stated. No other tuning is performed, and the same τ is used for all three tasks.

E.2 ANN-TO-SNN CONVERSION BASELINES

These methods share our setting most closely: they start from a trained ANN and produce an SNN without direct SNN training.

- **ReLU-Norm** (Rueckauer et al., 2017) (with the max-norm variant of): converts an *unbounded*-ReLU ANN by normalizing each layer by a scale λ_l estimated on a calibration set, so that weights and biases become $\tilde{W}_l = W_l \lambda_{l-1}/\lambda_l$ and $\tilde{b}_l = b_l/\lambda_l$. We implement both the max-norm ($\lambda_l = \max a_l$) and the robust percentile variant, normalizing one layer at a time and re-measuring after each so that the rescaling is correct when a normalization layer sits between the affine layer and the activation. We report the better of the two.
- **RTS** (Deng & Gu, 2021): threshold balancing plus a single closed-form bias shift $b_l \leftarrow b_l + \theta/(2T)$ that minimizes the half-step mismatch between the ANN's clipped activation and the SNN's step response. Its source activation is the threshold-ReLU, which is exactly ReLU1 at threshold 1, so no retraining is required. Note that T enters the stored parameters, so a converted model is tied to the tick budget it was converted for.
- **RMP-SNN** (Hao et al., 2023a): soft reset ($V \leftarrow V - V_{\text{th}}$) instead of hard reset, together with a layer-by-layer threshold balancing $V_{\text{th}}^l \leftarrow \alpha \max_l$ with $\alpha \leq 0.6$, each layer frozen before the next is estimated.
- **Spike-Calib** (Li et al., 2021): post-hoc calibration of the converted IF-SNN. We implement both the *light* (bias) and *advanced* (bias, initial potential and weights) variants, and follow the paper's greedy Algorithm 1: the T-step simulation is re-run before each layer is calibrated, so every layer is corrected against the signal it actually receives from the already-corrected layers upstream. BatchNorm folding (their Eq. 19) is already performed by our shared conversion front-end.
- **QCFS** (Bu et al., 2023): a training-time method. The source ANN is trained with the quantization-clip-floor-shift activation $a^l = \lambda^l \,\text{clip}\big(\frac{1}{L}\lfloor z^l L/\lambda^l + \phi \rfloor, 0, 1\big)$ with a trainable per-layer λ^l and the paper's recommended shift $\phi = 1/2$, and is then converted by $\theta^l \leftarrow \lambda^l$, $v^l(0) \leftarrow \theta^l \phi$. We train a separate ANN per task with $L = 8$.
- **SlipReLU** (Jiang et al., 2023): also training-time. The activation is the convex combination $c\,\theta\,\text{clip}(z/\theta + \delta_1, 0, 1) + (1-c)\,\theta\,\text{clip}(\frac{1}{N}\lfloor Nz/\theta + \delta \rfloor, 0, 1)$, converted by $V_{\text{th}} \leftarrow \theta$, $v(0) \leftarrow V_{\text{th}}\delta$. We use the authors' best reported settings $\delta_1 = 0$, $\delta = 1/2$, $c = 0.4$, and train a separate ANN per task.
- **DCGS** (Huang et al., 2025): the multi-threshold (MT) neuron, carrying $2n$ thresholds $\lambda_p = \theta/2^{p-1}$ for $p = 1, \ldots, n$ together with their negatives, and emitting at most one graded spike per time step whose magnitude is the λ_p closest to the membrane potential; $n = 1$ degenerates to an IF neuron with an added negative threshold. We use $n = 4$, and report the reported row under rate, since differential coding failed to achieve meaningful

accuracy on all tasks. Two threshold settings are run and we report the better: the per-layer threshold iteration of their Algorithm 1, with the fitted thresholds passed per site rather than collapsed to a single global value, and the fixed $\theta = 1$ that a bounded activation admits by construction.

- **AdaFire** and **AdaFire+SSC** (Wang et al., 2025): an adaptive-firing neuron emitting up to ϕ spikes per step, so its per-step output range is $[0, V_{\text{th}}\phi]$, with ϕ chosen per layer on the paper's sensitivity/energy criterion. **+SSC** adds Sensitivity Spike Compression, which raises the per-layer threshold by a factor ρ so that consecutive spikes merge into fewer, higher-amplitude ones. We do not implement the third component (Input-aware Adaptive Timesteps), an entropy-based early exit over a *bounded* sample, because this evaluation scores a perpetual stream at a fixed per-frame budget.
- **LIF-Reset**: our own converted LIF-SNN, run under the conventional protocol – membranes and codec states are reset at every frame boundary and the frame is simulated for T ticks. This is the direct ablation of streaming: identical weights, identical neurons, identical decoder, the reset being the only difference.

E.3 Directly-trained SNN baselines

These methods train a spiking network end-to-end with surrogate gradients. For methods that propose specific spiking neural architectures, we use the authors' released architectures and their released weights except DVS-Gesture task. For the DVS-Gesture task, we train all the models with our task setup.

- **Training rules.** Rectangular/boxcar surrogate gradients (STBP, Wu et al., 2018); the online rules OTTT (Xiao et al., 2022) and SLTT (Meng et al., 2023), whose memory is constant in T; the TET loss (Deng et al., 2022), which scores every tick rather than the time average; and ASGL (Wang et al., 2023), whose analog branch is itself a bounded ReLU.
- **Spike-driven Transformer (SDT)** (Yao et al., 2023) (gesture): spike-driven self-attention (SDSA), which replaces softmax attention with a Hadamard product and column-wise sums of binary spike matrices, with membrane shortcuts between blocks. We use a depth-4, width-256 skeleton of 4.15M parameters, comparable to GestureNet (3.96M), train it with the same stream windows, augmentation and EMA schedule as our reference ANN, and report the EMA checkpoint. Its forward pass steps through the 16 frames of a window, so its time steps are tied to its frame binning and we report it at this design point, $T = 16$. Re-binning a window into $T \neq 16$ frames, with the input rescaled to the trained per-frame intensity, moves the model off its training distribution and lowers accuracy on both sides (21.9–82.6 for $T \in \{2, \ldots, 64\}$). Spikformer (Zhou et al., 2022) and Meta-SpikeFormer (Yao et al., 2024), trained identically on the same skeleton, reached 89.4 and 88.8; we list the Spike-driven Transformer, which ties Spikformer overall and is slightly more accurate on the gesture classes (11 classes) (91.3 vs. 91.2).
- **EMS-YOLO** (Su et al., 2023) (detection): an ordinary binary LIF neuron ($V[t{+}1] = \tau V[t](1 - X[t]) + \sum_j W_{ij} X_j[t{+}1]$, $X[t{+}1] = H(V[t{+}1] - V_{\text{th}})$) with a rectangular surrogate, in the EMS-ResNet block that places a LIF before every convolution – including on the shortcut – so that every convolution receives a binary tensor. We evaluate the authors' COCO weights.
- **SpikeYOLO** (Luo et al., 2024) (detection): the I-LIF neuron $\text{clip}(\text{round}(U), 0, D)$ with the meta SNN blocks, trained with integer values and deployed with each integer step expanded into D binary sub-steps. We evaluate the authors' COCO weights in both the integer and the expanded-binary inference modes.

E.4 Common evaluation protocol

Shared reference ANN. Every post-hoc conversion baseline whose precondition our ANN satisfies is converted from the *same* trained checkpoint as LIF-Stream, so that the comparison prices what each method loses in conversion rather than differences in the source model. Methods whose activation function is part of their contribution (QCFS, SlipReLU) require their own source ANN; we train one per task with the identical architecture, schedule, and augmentation, changing only the activation. Methods that need unbounded ReLU statistics (ReLU-Norm, Spike-Calib, AdaFire) are given an unbounded-ReLU ANN trained the same way, plus the calibration split they require.

Reset convention. Every baseline is run under the protocol its paper specifies, which for all conversion baselines means a per-frame reset: the network state is cleared, the frame is simulated for T ticks, and the prediction is decoded. LIF-Stream is run without any reset after the start of the stream. The *LIF-Reset* row isolates this single difference on our own converted network. In Figure 6 and Section I.1, we additionally report what happens when the reset is simply removed from the published conversions.

Compute. All models we train ourselves are trained and evaluated on a single NVIDIA RTX 2080 Ti (12 GB).

F Zero-shot Transfer for Streaming Object Detection

F.1 Datasets

COCO 2017 (training). The reference ANN is trained for 80-class object detection on COCO 2017 (Lin et al., 2014a): 118,287 `train2017` images and 5,000 `val2017` images. We use only the box annotations. COCO is the source domain and is used for training and for reporting the ANN's frame-wise quality; no streaming evaluation is performed on it.

Argoverse-HD (streaming evaluation). Streaming detection is evaluated on Argoverse-HD (Li et al., 2020a), which pairs the front-center ring camera of Argoverse 1.1 (Chang et al., 2019) with dense 2D box annotations at 30 fps. We use the full validation split: 24 sequences and 15,062 annotated frames, ~ 627 frames per sequence on average, at a native resolution of 1920×1200. Argoverse-HD annotates eight categories – *car, truck, bus, person, bicycle, motorcycle, traffic light, stop sign* – all of which are COCO categories with the same definitions, so a COCO-trained detector is scored directly, with no fine-tuning and no modification of the detection head. The annotations are dense and are *not* temporally interpolated, so every frame is an independent ground-truth observation.

Why this pair. We chose a zero-shot transfer setting for two reasons. First, COCO is the benchmark on which SNN detectors are trained and compared, but it has no streaming evaluation; Argoverse-HD provides a rigorous 30 fps streaming benchmark but has no SNN detection literature. Second, the two share their label space, so transfer requires no head surgery and no retraining, and the comparison therefore measures streaming inference rather than adaptation. The same argument is made by Li et al. (2020a), whose own baselines are likewise off-the-shelf COCO models evaluated without training on Argoverse. We report ordinary per-frame COCO AP rather than the streaming-AP metric of Li et al. (2020a): sAP scores a detector against the world at the moment its output is emitted and therefore requires a wall-clock latency model, whereas the quantity under study here is the tick budget a frame needs, which is a hardware-independent count.

F.1.1 Licenses

COCO images are made available under the Flickr Terms of Use and its annotations under the Creative Commons Attribution 4.0 License. Argoverse 1.1 is released by Argo AI under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 License, and the Argoverse-HD annotations are released by Li et al. (2020a) for non-commercial research use. All datasets are used within their licensed terms, for non-commercial academic research.

F.2 Model

The reference ANN is a convertible YOLOv8-style detector (Jocher et al., 2023): a CSP backbone, a PAN–FPN neck, and a DFL detection head with task-aligned assignment, at three pyramid levels with strides $(8, 16, 32)$. We use the `v8m`-scale configuration – stage widths $(48, 96, 192, 384, 576)$ and backbone depths $(2, 4, 4, 2)$ – giving 25.05M parameters and 39 sequential spiking layers along the deepest path.

Three substitutions make the model convertible, and each one is applied *before* training, so the trained ANN and the converted SNN describe exactly the same function:

- **SiLU → ReLU1.** Every activation in the backbone, neck and head is the bounded ReLU1, which is what lets the converted network use a fixed unit threshold with no data-driven calibration.
- **MaxPool → AvgPool.** The SPPF block's max-pooling is the only operator in the architecture with no affine equivalent; we use average pooling in its place. (An exact reconstruction of max-pooling from ReLU1 units is also available and gives the same function at a higher layer count; we do not use it in the reported model.)
- **Post-processing kept outside the graph.** Sigmoid decoding, DFL integration, coordinate de-normalization and NMS are parameter-free and run on the decoded output, exactly as in the ANN, and therefore never enter the converted network.

Baselines that require a different activation (QCFS, SlipReLU) or an unbounded one (ReLU-Norm, Spike-Calib, AdaFire) are trained as separate arms of the same architecture, schedule and augmentation, changing only the activation; the directly-trained SNN detectors (EMS-YOLO, SpikeYOLO) use their authors' published architectures and weights.

F.3 Training Setup

Training follows the standard YOLOv8 recipe, adapted only where the convertibility constraint requires it. The model is trained from scratch on COCO `train2017` for 300 epochs at an input resolution of 640, with SGD (learning rate 0.005, final learning-rate factor 0.01, Nesterov momentum 0.937, weight decay 5×10^{-4}) and a cosine schedule, after a warm-up of 10,992 optimizer steps (≈ 3 epochs) over which the momentum is ramped from 0.8. The effective batch size is 32, realized as batch 16 with two gradient accumulation steps. Augmentation is mosaic ($p = 1.0$, disabled for the final 10 epochs), horizontal flip ($p = 0.5$), HSV jitter $(0.015, 0.7, 0.4)$, and random affine applied on the composed mosaic canvas with translation 0.1 and scale 0.5. The loss is the YOLOv8 composite – CIoU box loss, distribution focal loss, and BCE classification loss – with gains $(7.5, 1.5, 0.5)$ and task-aligned assignment. We use mixed precision, TF32 matmuls, and an EMA of the weights; the EMA weights are the ones converted. Checkpoints are selected on `val2017` mAP.

F.4 Evaluation Setup

Input pipeline. Each frame is letterboxed to 640×640, scaled to $[0, 1]$ and normalized with the ImageNet statistics used during training. Frames are presented in their recorded order at the native 30 fps and no frame is skipped, since the inter-frame interval is the quantity the streaming claim depends on.

Streaming protocol. For LIF-Stream, the converted SNN is instantiated once per sequence, its membrane and codec states are initialized once, and the 24 sequences are then processed frame by frame with T ticks per frame. The prediction for a frame is read at the last tick allotted to that frame. For the reset baselines, the state is cleared at every frame boundary and the frame is simulated for T ticks, which is the protocol every published conversion method specifies. The dense ANN is run on the same frames .

Metric. We report COCO AP@50 computed with `pycocotools`, with a score threshold of 0.001, NMS IoU 0.7 and at most 300 detections per frame, which is the standard COCO convention. The dense ANN and the streaming arms are scored over all 15,062 annotated frames of the validation split; the per-frame-reset conversion baselines are scored on the same 24 sequences under the protocol their own papers specify. Every arm is scored by the same metric code, and each comparison reported in the paper is made over one common frame set. Efficiency is reported as described in Section E.4.

G Streaming Automatic Speech Recognition (ASR)

G.1 Datasets

We train and evaluate on LibriSpeech (Panayotov et al., 2015), the standard English read-speech corpus derived from LibriVox audiobooks and sampled at 16 kHz. Training uses the full 960 h partition (`train-clean-100`, `train-clean-360`, `train-other-500`), 280,663 utterances

totalling 958.2 h after dropping utterances longer than 17 s. `dev-clean` is used for validation and model selection, and `test-clean` and `test-other` are used for the reported word error rates.

Unlike the usual offline protocol, which presents a whole pre-segmented utterance to the model at once, we stream each utterance to the model in fixed-size chunks and require a prediction after every chunk, so that the model never sees future audio beyond its own bounded look-ahead.

G.1.1 LICENSES

LibriSpeech is distributed under the Creative Commons Attribution 4.0 License, and is used here for non-commercial academic research.

G.2 MODEL

The reference ANN is a windowed SummaryMixing Conformer encoder (Parcollet et al., 2024) with a CTC head, adapted so that the whole encoder is a composition of affine operators and ReLU1. SummaryMixing is the natural starting point for a convertible streaming encoder: it replaces self-attention's data-dependent $T \times T$ mixing with a *constant* one – a uniform mean over time – which is an affine operator, whereas softmax attention is not.

Front-end. 80-bin log-mel filterbank features at a 10 ms hop, normalized with a *fixed global* CMVN fitted once on the training set and frozen (per-utterance CMVN is neither causal nor foldable into the network), followed by a convolutional subsampling of $\times 4$, giving one encoder frame every 40 ms.

Window assembly. The encoder consumes one window per call, formed by concatenating the previous chunk and the current chunk, $[\,\text{prior chunk} \,\|\, \text{current chunk}\,]$. This is the design decision that keeps the encoder feed-forward: the published streaming SummaryMixing variant accumulates its summary statistic *across* calls, which is a cross-call recurrence, whereas taking the mean over a two-chunk window makes the same summary a reduction *inside* a single forward pass – an ordinary `AvgPool1d`. With a chunk of 32 encoder frames and a left context of 32 frames, the window spans 2.56 s and the model emits a 1.28 s chunk of frame posteriors per call. The first window of a stream is zero-padded on the left and its mean is renormalized accordingly.

Block. Each of the 18 encoder blocks applies two per-frame branches $f(X)$ and $s(X)$ (Linear + ReLU1), averages $s(X)$ over the window with a single `AvgPool1d` whose kernel equals the window length, broadcasts the resulting summary back to every frame, concatenates it with $f(X)$, and projects with a further Linear + ReLU1. A depthwise convolution module (kernel 31 or 63) and a feed-forward sandwich complete the block, as in a Conformer. The model dimension is 256; the encoder has $\approx$25–28M parameters and 128–182 spiking sites depending on the arm.

Substitutions for convertibility. GELU/Swish are replaced by ReLU1; LayerNorm by masked BatchNorm, since LayerNorm computes per-sample statistics at run time and therefore cannot be folded into the preceding affine layer; and the streaming variant's cumulative masked mean by the windowed `AvgPool1d` described above. The head is CTC, as in the source recipe; CTC collapse and beam search run on the decoded posteriors, outside the converted graph.

G.3 TRAINING SETUP

The encoder is trained with CTC on the 960 h partition for 40–53 epochs, with Adam (peak lr 8×10^{-4}, 5k–15k warm-up steps, then inverse-square-root decay), dropout 0.1, weight decay 5×10^{-4}, gradient clipping, and mixed precision. The effective batch size is 14 utterances $\times$ gradient accumulation, with a length-bucketed sampler. Augmentation is speed perturbation plus DropFreq/DropChunk on the waveform, and SpecAugment on the features where indicated. We keep an EMA of the weights (decay 0.9999) and average the final 10 epoch checkpoints, as is standard for this recipe. Two vocabularies are used: a 29-symbol character set and a 5000-token unigram sub-word model; and some arms add an auxiliary attention decoder to the CTC loss during training only. *The auxiliary decoder is discarded before conversion and is never part of the deployed or converted model*; it acts purely as a training-time regularizer. Training goes through exactly the same window assembly

that inference uses, so the model is never trained on context it will not have at deployment; this equivalence is asserted by a unit test on every commit.

G.4 Evaluation Setup

Streaming protocol. Each utterance is streamed chunk by chunk with 40ms inter-chunk interval. For LIF-Stream, the converted SNN's state is initialized once at the beginning of the utterance and then persists across chunk boundaries; each chunk is allotted T ticks and the frame posteriors for that chunk are read at its final tick. The first chunk optionally receives a warm-up of extra ticks. For the reset baselines the state is cleared at every chunk boundary, which is the protocol those methods assume. We use $\tau = 8$ throughout, as on the other two tasks.

Decoding and metric. Frame posteriors are decoded with CTC greedy decoding, or with a CTC prefix beam search combined with an n-gram language model where stated; decoding is identical for every arm, so it never contributes to a difference between arms. We report corpus-level word error rate (WER) on `test-clean` and `test-other`.

H Gesture Recognition on Event Streams

H.1 Datasets

We use DVS128 Gesture (Amir et al., 2017a), the standard neuromorphic gesture benchmark, recorded with a DAVIS128 event camera at 128×128 spatial resolution. It contains $1{,}342$ gesture instances over 11 classes (10 gestures plus an *other* class), performed by 29 subjects under three lighting conditions, and distributed as 122 *continuous* recordings with interval labels. We use the official subject split: 98 recordings for training and 24 for test.

Streaming, not clip-wise. Most recent work evaluates this dataset clip-wise: each pre-segmented gesture is cut out in advance, binned by constant event count, and classified as a whole. That protocol is not causal – constant-count binning needs the end of the clip to place its bins, and cutting the clip out presupposes that the gesture has already been segmented. We therefore evaluate on the *unsegmented* recordings, as the original benchmark does (Amir et al., 2017a), and require the model to predict on the fly from the events it has received so far. Concretely, we slide a 2 s window over each recording and score every window position.

A 2 s window is the geometry that follows from the data. Gestures last 6.53 s on average and the gaps between them 1.84 s, so a window as long as a gesture straddles two gestures in over half of all positions, whereas a 2 s window is clean in 56.3% of positions and spans two gestures in only 7.4%. Because the gestures are repetitive, a 2 s window still contains complete motion cycles. Since a streaming evaluation also encounters rest periods, the stream task carries a 12th *null* class.

H.1.1 Licenses

The DVS128 Gesture dataset is released by IBM Research under the Creative Commons Attribution 4.0 License. We use the distributed archive as published, verified by its MD5 checksum, and apply only the corrected label file for one recording released with the dataset mirror (one gesture's end time is recorded after the next gesture's start time; the corrected file resolves the overlap). The dataset is used for non-commercial academic research.

H.2 Model

The reference ANN is *GestureNet*, a convertible TCN-style network of 3.96M parameters with 22 spiking sites:

stage	operator
input	$[B, T, C, 128, 128]$ event histogram
stem	5×5 conv, stride $2 \rightarrow$ BN $\rightarrow$ ReLU1
spatial backbone	4 stages of residual blocks, AvgPool2d(2) between stages
spatial pooling	AvgPool2d with a kernel fixed at build time
temporal	2 `Conv1d` blocks over the frame axis
readout	AvgPool1d$(T) \rightarrow$ Linear

Two design decisions carry the model. First, the spatial backbone is applied frame-wise with shared weights, with T folded into the batch dimension, so the number of spiking sites is $O(\text{depth})$ rather than $O(\text{depth} \times T)$ – which is what makes a multi-frame window affordable at a small tick budget. Second, the time axis is mixed by convolutions rather than by a mean: four of the eleven classes are chirality pairs (left/right hand, clockwise vs. counter-clockwise) that differ only in temporal order, and any order-invariant reduction maps a gesture and its time-reverse to the same representation. AvgPool1d is retained, but only after the temporal convolutions have made the representation order-sensitive. The head is a bare linear layer with no terminal activation. The same constraint rules out horizontal flip and temporal reversal as augmentations: both map a class onto its partner's label.

Input representation. Events are accumulated into a voxel grid, without normalization or clamping. For each event (t, x, y, p) falling in the window, the bin index is $\lfloor (t - t_{\text{start}})/\Delta \rfloor$ and the corresponding cell $[\text{bin}, p, y, x]$ is incremented. The 2 s window is divided into 16 equal bins of 125 ms, giving $[16, 2, 128, 128]$ for the two event polarities. Beside those disjoint bins we carry a running mean of them, $x_{\text{cum}}[k] = \frac{1}{k+1} \sum_{j \le k} x[j]$, concatenated along the channel axis to give $[16, 4, 128, 128]$. The two halves hold different things and are kept together rather than one replacing the other: the disjoint bins carry instantaneous ON/OFF polarity structure, which is what separates the rotation classes, while accumulation greys that structure out as both polarities pile onto the same pixels; the running mean in turn carries the longer-range shape of the gesture, which no single 125 ms bin contains. The cumulative sum is an invertible linear map along the frame axis, so nothing is lost relative to the disjoint bins and the traced graph stays affine + ReLU1 – the only structural change is that the stem takes 4 input channels instead of 2.

H.3 Training Setup

The model is trained on 2 s windows drawn from the training recordings at random offsets, so that every training window is distinct. We use AdamW with a OneCycle schedule, label smoothing 0.1, dropout 0.1, gradient clipping at 1.0, and standard BatchNorm initialization, for 200 epochs. Augmentation is random spatial roll and rotation only, for the chirality reason above. We keep an EMA of the weights and evaluate both the raw and the EMA checkpoints. The reported configuration is run with 3 seeds; the measured seed spread is 0.51 pp, below the eval-to-eval oscillation within a single run. Since one test clip is 0.347 pp on the 288-clip protocol, we treat differences below $\sim$1.4 pp as not meaningful and do not draw conclusions from them.

H.4 Evaluation Setup

Streaming protocol. Each test recording is streamed as a sequence of 2 s windows at a 0.25 s stride. For LIF-Stream, the converted SNN's state is initialized once per recording and persists across windows, with T ticks per window; the prediction is read at the final tick of each window. Reset baselines clear their state at every window. A window is labelled by the class at its center; windows falling in a rest period take the *null* label. We use $\tau = 8$, as on the other two tasks.

Metrics. The primary metric is per-window classification accuracy on the stream. We additionally report two protocols that connect to the literature. (i) *Per-decision accuracy in the style of the original benchmark* (Amir et al., 2017a): decisions are taken on a sliding window at a 0.1 s stride at every position inside a labelled gesture, the null column is dropped, and the arg-max is taken over the 11 gesture classes, with an optional majority-vote horizon standing in for their sliding-window filter; we also report the 10-category variant that excludes *other*, mirroring their own two numbers. (ii) *Clip accuracy* on the standard 288-clip test protocol, obtained by sliding the same 2 s windows inside each

test clip, averaging the logits and taking the arg-max over classes 1–11; this is the number comparable to the published leaderboard. Efficiency is reported as in Section E.4.

I Experimental Setups for the In-depth Analyses

All four studies in the in-depth analysis are run on Argoverse-HD with the detector and the streaming protocol described in Section F, at 640 input resolution and $\tau = 8$. Unless stated otherwise they use the full validation split (24 sequences, 15,062 frames) and never stride the stream, since the inter-frame interval is the independent variable in two of the four studies.

I.1 Integration with IF conversion methods

Question. Our framework and the advanced IF conversion methods optimize different things – ours makes the trajectory track a moving reference across frames, theirs makes a single bounded simulation more accurate – so it is worth asking whether the two compose.

Porting. For each published conversion method we take the converted network exactly as that method produces it, using the same implementation that produces its row in Table 1, and run it under two protocols that differ by exactly one call: the method's own per-frame reset, and our streaming protocol in which the state is initialized once per sequence and then persists. Nothing else changes – same weights, same calibration, same decoder, same frames, same score threshold – so any difference is attributable to the reset alone. Methods that pre-charge each neuron to a fixed initial potential (e.g. $\theta/2$ for QCFS and SlipReLU) store that value as the neuron's *reset* value, so it is re-applied by the reset call and only by the reset call; AdaFire likewise clears its burst state per frame.

Measurement. All arms are scored over one identical frame set – whole sequences, never strided – with AP@50 at a score threshold of 0.001, and sweep $T \in \{8, 16, 32, 64, 128\}$ (plus $T = 256$ for the IF arm, whose published protocol needs a far larger budget). The dense ANN on the same frames provides the ceiling. Because the per-frame-reset baselines and the streaming arm are scored on the same frames, the ratios are meaningful even where the absolute values differ from the full-validation numbers of Table 1. Some of these baselines require a source ANN trained with their own activation, or an unbounded-ReLU network plus calibration data, so the comparison necessarily crosses checkpoints; this is structural rather than a choice, and is itself part of the point, since a bounded activation needs nothing fitted.

I.2 Inter-frame prediction efficiency

Question. A uniform budget charges every frame the same T. If frames differ in how many ticks they actually need, the distribution of the *needed* budget is the quantity the efficiency claim should be stated in, and the gap between it and a flat T is what incremental computing buys.

Defining "settled". The settle tick of frame k is measured on the *detections*, not on the raw readout:

$$A_k[t] = F_1\big(\mathrm{dets}(y_t),\, \mathrm{dets}(y_{T_{\max}})\big), \qquad \mathrm{settle}(k) = \min\{\, t : A_k[t'] \geq 1-\epsilon \text{ for all } t' \geq t \,\}, \quad (48)$$

where detections are matched by class at IoU ≥ 0.3 and F_1 is computed over detections above a confidence of 0.3. This is what "the answer has stopped changing" means for a detector, and it is insensitive to the slow residual drift that a threshold on the float readout would pick up. The stability threshold (0.3) is deliberately distinct from the scoring threshold (0.001, the COCO convention): at the scoring threshold the output carries a long tail of low-confidence boxes whose ordering never stabilizes. $T_{\max} = 64$ is used as the reference budget.

Complexity label. The per-frame complexity on the x-axis is computed *independently of the tick count*, otherwise the plot would be a tautology. We use the dense ANN's own frame-to-frame output change, $\mathrm{drift}(k) = \|y_{\mathrm{ANN}}(k) - y_{\mathrm{ANN}}(k{-}1)\| / \|y_{\mathrm{ANN}}(k{-}1)\|$, measured in the same flattened head space; the ANN arm is run anyway as the accuracy ceiling, so this costs nothing extra. Raw pixel change is reported as a secondary, interpretability-only label.

I.3 Finer temporal resolution helps streaming SNNs

Question. Tick cost is normally quoted per frame, which silently treats the frame rate as given. It is a design choice, and the two protocols respond to it in opposite ways: with a per-frame reset every frame pays the full transient, so cost per second is $T \times$ fps, exactly linear in the frame rate with no property of the video entering; with a persisting state a higher frame rate means a smaller change between consecutive frames, so each frame needs fewer ticks.

Setup. We re-use the settle-tick definition of Section I.2 unchanged – same detection-space criterion, same matching threshold, same ϵ grid – and sweep the sampling rate by striding the source video: strides $1, 2, 3, 4, 5, 6, 10$ of the native 30 fps stream give $30, 15, 10, 7.5, 6, 5, 3$ fps. Each point is the full validation split at that rate. Cost per second of video is fps $\times$ $\overline{\text{settle}}$(fps). The reset arm needs no run: its frames are independent, so its cost is $T \times$ fps analytically, and we use the tick budget at which it attains the same accuracy as the streaming arm on the same data. Striding is what the streaming protocol otherwise forbids, precisely because it widens the inter-frame gap and removes the redundancy persistence exploits; here that is the independent variable. Consequently the numbers in this study are comparable only across its own sweep, the 30 fps column being the one it shares with the main streaming results.

Reporting. We fit a power law to cost per second against frame rate for each arm and report the exponent, together with the censored fraction at every point; points whose censoring is large enough to understate the mean are excluded from the figure rather than reported with a caveat. The claim the measurement supports is the ratio of the two exponents – reset grows linearly with frame rate by construction, while the streaming arm grows sublinearly – so the advantage widens as the stream is sampled more finely.

I.4 Qualitative visualization of the inference trajectory

For Figure 5 we project the dense ANN's per-frame outputs onto their own top two principal directions and place every SNN tick of every arm on that same basis, so that all arms are shown in one coordinate system defined by the reference model rather than by any of the spiking arms. The projection is computed on a single Argoverse-HD sequence, with the detector and protocol of Section F; distances quoted in the text are measured in that 2-D projection between an arm's final tick for a frame and that frame's ANN target. The arms shown are the published IF conversion protocol, the LIF conversion under a per-frame reset, and our LIF-Stream, each at its own tick budget, so that the figure shows what a given budget buys rather than a budget-matched comparison alone.

I.5 Empirical Validation of our theory in Figure 1

We use a small, fully controlled network. The reference ANN is an MLP of seven affine layers of width 64 (12-dimensional input, 16-dimensional output, 22.7K parameters) whose only nonlinearity is ReLU1. It is trained on a teacher–student task in which inputs x and targets y are generated from a shared two-dimensional latent z: the input is a random linear lift of the latent into 12 dimensions followed by a tanh squash, and the target is a bank of sinusoids of independent random projections of that same latent, each with its own phase.

$$\mathbf{z} \sim \mathcal{N}\big(\mathbf{0},\, 0.8^2\mathbf{I}_2\big) \qquad \text{shared latent} \tag{49}$$

$$\mathbf{x} = 0.5 + 0.4\,\tanh(\mathbf{A}\mathbf{z}) \in (0.1, 0.9)^{12} \qquad \text{network input} \tag{50}$$

$$\mathbf{y} = 0.5 + 0.45\,\sin(\mathbf{W}\mathbf{z} + \boldsymbol{\varphi}) \in (0.05, 0.95)^{16} \qquad \text{regression target} \tag{51}$$

where $A_{ij} \sim \mathcal{N}(0, \frac{1}{2})$, $W_{ij} \sim \mathcal{N}(0, 1.6^2)$ and $\varphi_i \sim \mathcal{U}(0, 2\pi)$ are sampled once and held fixed, and $\tanh$, $\sin$ act elementwise. Training is 1.5K Adam steps (batch 256, learning rate 3×10^{-3}, fixed seed). The evaluation input stream is the latent traversing a Lissajous curve over 24 frames. We convert the network with LIF neurons and feed the input 32 ticks per frame in a single streaming pass, and membrane state persists across the entire 768-tick stream. All distances are measured in the full 16-dimensional readout space, with PCA used only for visualization.