INCORPORATE SELF-PREDICTION TO ENHANCE DYNAMICS OF SPIKING NEURONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Spiking Neural Networks (SNNs) offer the potential for energy-efficient computation through event-driven, sparse processing, but their training is challenged by spike non-differentiability and trade-offs among performance, efficiency, and biological plausibility. Predictive coding motivates exploring how a neuron's own input–output history can modulate its subsequent dynamics. Inspired by this, we propose a self-prediction enhanced spiking neuron framework that generates an internal prediction current from its input–output history to modulate membrane potential, with separate coefficients for input memory and spike feedback. This design introduces additional gradient pathways through the auxiliary state and provides a local feedback mechanism inspired by dendritic modulation and neuronal self-feedback. Experiments with the reference feedback rules show performance gains in most evaluated configurations across diverse architectures, neuron types, time steps, and tasks. LIF ablations show that learning the spike-feedback coefficient further improves accuracy in two architectures.

1 INTRODUCTION

Spiking Neural Networks (SNNs) constitute a biologically plausible class of neural network models that communicate through discrete, time-encoded spikes (Maass, 1997). Unlike conventional Artificial Neural Networks (ANNs), which rely on continuous activation values, SNNs activate only when a neuron's membrane potential crosses a firing threshold (Gerstner et al., 2014). This event-driven mechanism leads to highly sparse computation, offering opportunities for energy-efficient deployment on resource-constrained edge hardware (Li et al., 2024). Recent neuromorphic platforms such as Loihi (Davies et al., 2018), TrueNorth (DeBole et al., 2019), Darwin (Ma et al., 2017) and Tianjic (Pei et al., 2019) have demonstrated the potential of SNNs to enable low-power, real-time inference. Despite these benefits, training high performance SNNs remains a big challenge. The core issue stems from the binary, non-differentiable nature of spike events, which is not suitable for standard backpropagation. Moreover, simulating temporal dynamics during training often demands substantial memory and computational overhead.

To circumvent these limitations, researchers have developed two main paradigms for building high-performance SNNs: direct training and ANN-to-SNN conversion. Direct training approaches attempt to learn SNN weights end-to-end by approximating gradients through surrogate gradient functions (Neftci et al., 2019; Duan et al., 2022), or by leveraging biologically motivated rules like spike-timing-dependent plasticity (STDP) (Masquelier & Thorpe, 2007; Masquelier et al., 2008; 2009). While these methods aim to fully exploit the temporal information of spiking dynamics, they often suffer from long convergence times, and accuracy gaps on benchmark datasets (Gygax & Zenke, 2025). In contrast, ANN-to-SNN conversion first trains a high-accuracy ANN using standard deep learning methods and then converts it into an SNN that approximates its input–output behavior through rate-based or temporal coding schemes (Cao et al., 2015; Rueckauer et al., 2017; Han et al., 2020; Li et al., 2021; Bu et al., 2022; Huang et al., 2024b). Although conversion can exploit event-driven inference, it usually incurs a non-negligible accuracy drop especially under tight latency. Meanwhile, the resulting SNN is trained primarily to approximate the ANN's input–output mapping, motivating complementary approaches that explicitly shape neuronal dynamics.

Beyond these training advances, predictive coding provides a motivation for incorporating history-dependent feedback into spiking neurons. Evidence shows that individual auditory neurons fire

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.

selectively to omitted expected sounds, directly signaling prediction errors at the single-neuron level (Lao-Rodríguez et al., 2023; 2026). Pre-activated neural templates resembling actual stimulus responses further support this predictive coding framework (SanMiguel et al., 2013), with recent work identifying neurons specifically tuned to negative prediction errors (Yaron et al., 2025). Beyond population-level mechanisms, prediction may also occur within single neurons. Hawkins & Ahmad (2016) proposed that dendritic branches act as independent pattern detectors, enabling internal predictions. This aligns with models where dendrites predict somatic activity, and local prediction errors drive synaptic plasticity (Urbanczik & Senn, 2014; Brea et al., 2016; Sacramento et al., 2018). These observations motivate studying how input and output history can jointly modulate a spiking neuron's subsequent activity.

Inspired by these observations, we propose a self-prediction enhanced spiking neuron method applicable to multiple spiking neuron models and network architectures. The core idea is to enable each neuron to leverage its own historical inputs and outputs to form a weighted input–output residual and integrate it into a prediction current, which dynamically modulates its membrane potential as an auxiliary input current. This prediction current retains input and output history and feeds back into subsequent membrane integration, thereby facilitating or suppressing later spike emissions. Here, self-prediction denotes the history-dependent drive that modulates subsequent membrane integration. We introduce coefficients a^l and b^l to separate input memory from spike feedback, encompassing the reference neuron rules as special cases and allowing either contribution to be learned independently.

Our main contributions are summarized as follows:

- We propose a self-prediction current based on a weighted input–output residual, with separate coefficients for input memory and spike feedback, and show how it augments multiple spiking neuron models.
- We analyze the self-prediction enhanced neuron from the perspectives of dynamics, gradient propagation, and biological motivation. Single-neuron simulations illustrate the roles of the two history components.
- We evaluate our method on image classification tasks across various network architectures, spiking neuron models, and datasets. Coefficient ablations explore learning the input and spike-feedback strengths separately or jointly.
- We further validate our approach in sequential classification tasks and reinforcement learning tasks to assess its applicability beyond static image classification.

2 Related Works

2.1 Direct Training of SNNs

Recent years have witnessed significant progress in directly training deep SNNs. Pioneering works such as STBP (Wu et al., 2018), SuperSpike (Zenke & Ganguli, 2018), and SLAYER (Shrestha & Orchard, 2018) introduced surrogate gradient methods to avoid the non-differentiability of spike emission, enabling end-to-end supervised learning. Subsequent efforts further enhanced SNN performance through more expressive neuron models such as learnable membrane time constants (Fang et al., 2021b), adaptive firing thresholds with moderate dropout (Wang et al., 2022), complementary LIF dynamics to mitigate temporal gradient vanishing (Huang et al., 2024a), and unified gated LIF frameworks that integrate diverse biological features (Yao et al., 2022). Concurrently, architectural advances in residual architectures such as Spiking ResNet (Hu et al., 2023), SEW ResNet (Fang et al., 2021a), and MS-ResNet (Hu et al., 2025) have mitigated degradation and enabled stable training of deep networks; Meanwhile, attention-based models including Spikformer (Zhou et al., 2023), Spike-driven Transformer (Yao et al., 2023), and QKFormer (Zhou et al., 2024), have successfully integrated self-attention mechanisms into the spiking paradigm.

Recurrent SNNs enrich neuronal dynamics through feedback connections (Liang & Hu, 2015; Liang et al., 2015; Yin et al., 2020; Liao & Poggio, 2016). Within neurons, ALIF/LSNN uses spike-dependent threshold adaptation (Bellec et al., 2018), while adLIF couples membrane potential and spikes to an adaptation current (Baronig et al., 2025). STC uses previous spikes to multiplicatively modulate input and membrane history, with learned neighborhood interactions (Wang & Yu, 2024).

Our formulation accumulates weighted input and subtractive spike history in a neuron-local auxiliary state that enters the next integration additively. This design separates the strengths of the two history components while retaining the underlying membrane and reset rules.

2.2 Predictive Coding Theory

Predictive coding (PC) aims to minimize redundancy by propagating prediction errors rather than raw sensory signals (Elias, 1955). In neuroscience, PC has been formalized as a hierarchical inference framework where higher areas generate predictions and lower areas compute local mismatches (Huang & Rao, 2011; Spratling, 2017). A growing body of work indicates that prediction operates not only at the network level but within individual neurons (Hawkins & Ahmad, 2016). Consistent with this view, several computational models formalize dendritic processing as a form of predictive coding, where basal or apical dendrites learn to predict somatic activity, and synaptic plasticity is driven by local prediction errors (Urbanczik & Senn, 2014; Brea et al., 2016; Sacramento et al., 2018).

Predictive coding has also inspired several SNN models, such as PC-SNN (Lan et al., 2022), which uses firing time prediction errors for supervised learning, and SNN-PC (Lee et al., 2024), which employs dedicated positive and negative error neurons for unsupervised reconstruction. Yin et al. (2018) point out that autaptic connections formed by pyramidal cells, where the axon synapses onto the neuron's own dendrites or soma, are a key component of neural circuitry. Motivated by neuronal self-feedback, our formulation combines input and spike histories within a local auxiliary state and separates their contributions through two coefficients.

3 Preliminaries

For a Leaky Integrate-and-Fire (LIF) layer l at time step t, synaptic input, membrane integration, spike generation, and hard reset are

$$\boldsymbol{I}^l[t] = \boldsymbol{x}^{l-1}[t] = \boldsymbol{W}^l[t]\boldsymbol{s}^{l-1}[t], \tag{1}$$

$$\boldsymbol{m}^l[t] = (1 - \frac{1}{\tau^l})\boldsymbol{v}^l[t-1] + \frac{1}{\tau^l}\boldsymbol{I}^l[t], \tag{2}$$

$$\boldsymbol{s}^l[t] = \mathrm{H}(\boldsymbol{m}^l[t] - \theta^l), \tag{3}$$

$$\boldsymbol{v}^l[t] = \boldsymbol{m}^l[t] - \boldsymbol{s}^l[t](\boldsymbol{m}^l[t] - v_{reset}), \tag{4}$$

Here $\boldsymbol{W}^l[t]$ is the synaptic weight matrix; $\boldsymbol{I}^l[t] = \boldsymbol{x}^{l-1}[t]$ is the weighted input; $\boldsymbol{s}^l[t]$ is the spike vector; and $\boldsymbol{m}^l[t]$, $\boldsymbol{v}^l[t]$ are the membrane potentials before and after reset. The membrane time constant is τ^l, the firing threshold is θ^l, H is the Heaviside function, and v_{reset} is a fixed reset potential, typically zero.

We also evaluate Integrate-and-Fire (IF), Parametric LIF (PLIF), and Complementary LIF (CLIF) neurons. IF omits leakage, using $\boldsymbol{m}^l[t] = \boldsymbol{v}^l[t-1] + \boldsymbol{I}^l[t]$; PLIF learns τ^l; and CLIF adds a complementary membrane state that modulates reset. Soft reset uses $\boldsymbol{v}^l[t] = \boldsymbol{m}^l[t] - \boldsymbol{\theta}^l\boldsymbol{s}^l[t]$. Full variant equations and additional descriptions are given in Appendix.

4 Method

We formulate self-prediction enhancement as a local feedback framework (Fig. 1) that combines input memory and a neuron's own spike history. Two layerwise coefficients separate their contributions, while appropriate coefficient choices recover the reference neuron rules. The state is driven directly by weighted input and emitted spikes, and modulates subsequent membrane integration.

4.1 A General Local Feedback Framework

We retain the notation of Section 3: $\boldsymbol{x}^{l-1}[t]$ is the weighted input to layer l, $\boldsymbol{m}^l[t]$ and $\boldsymbol{v}^l[t]$ are its membrane potentials before and after reset, and $\boldsymbol{s}^l[t]$ is its output spike. The auxiliary state $\boldsymbol{m}_p^l[t]$ integrates input and output history. Starting from $\boldsymbol{m}_p^l[0] = \boldsymbol{0}$, each step first uses the previous

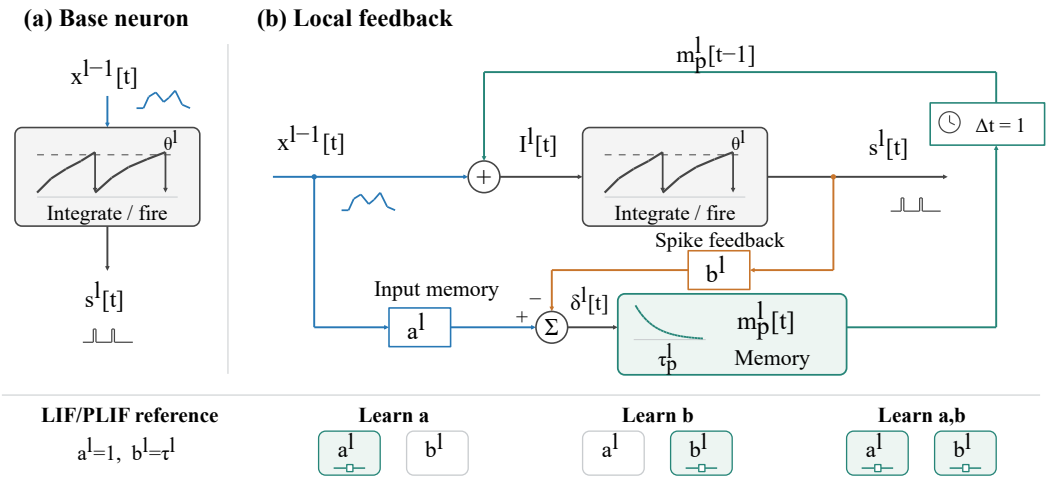


Figure 1: Local feedback framework. The base neuron (a) is augmented with input memory and subtractive spike feedback (b). Schematic icons denote membrane integration/reset, memory decay and a one-step delay. Sliders mark independently learnable coefficients. The reference setting shown is for LIF/PLIF. The state update follows Eq. equation 7; numerical settings and implementation conventions are specified in Appendix.

auxiliary state, then updates it after generating the current spike:

$$\boldsymbol{I}^l[t] = \boldsymbol{x}^{l-1}[t] + \boldsymbol{m}_p^l[t-1], \tag{5}$$

$$\boldsymbol{\delta}^l[t] = a^l \boldsymbol{x}^{l-1}[t] - b^l \boldsymbol{s}^l[t], \tag{6}$$

$$\boldsymbol{m}_p^l[t] = (1-\tau_p^l)\boldsymbol{m}_p^l[t-1] + \tau_p^l \boldsymbol{\delta}^l[t]. \tag{7}$$

Here $0 < \tau_p^l < 1$ sets memory retention, $a^l \geq 0$ weights input memory, and $b^l \geq 0$ weights spike feedback. These scalars are shared by the neurons within each spiking layer. The notation $\boldsymbol{\delta}^l[t]$ denotes the weighted input–output residual that drives the auxiliary update.

For hard-reset LIF, the augmented current in Eq. equation 5 enters the membrane integration in Eq. equation 2; spike generation and reset follow Eqs. equation 3 and equation 4. The auxiliary update creates the causal loop $\boldsymbol{s}^l[t] \to \boldsymbol{m}_p^l[t] \to \boldsymbol{m}^l[t+1] \to \boldsymbol{s}^l[t+1]$. The one-step delay determines the update order. For IF, PLIF and CLIF, we retain the corresponding membrane integration and reset rules and attach the same input–output memory update.

Reference rule and learnable extensions. The LIF/PLIF reference rule is the special case $a^l = 1$ and $b^l = \tau^l$:

$$\boldsymbol{m}_p^l[t] = (1-\tau_p^l)\boldsymbol{m}_p^l[t-1] + \tau_p^l \left(\boldsymbol{x}^{l-1}[t] - \tau^l \boldsymbol{s}^l[t]\right). \tag{8}$$

The framework also permits independently learning either coefficient or both. For positive initial values, we use $a^l = a_0^l \exp(\phi_a^l)$ and $b^l = b_0^l \exp(\phi_b^l)$, with zero-initialized ϕ_a^l, ϕ_b^l; fixed coefficients are excluded from optimization. Learning only b^l adds one scalar per spiking layer beyond the original learnable τ_p^l. It changes the balance between input memory and spike feedback without adding another recurrent state. For PLIF, $b^l = \tau^l$ follows the learned membrane time constant; independently learning b^l removes this coupling. IF and CLIF use $a^l = b^l = 1$ in the reference implementation. The code-to-notation mapping is given in Appendix.

4.2 Memory and Feedback Interpretation

For coefficients held fixed during one forward sequence, expanding Eq. equation 7 gives

$$\boldsymbol{m}_p^l[t] = \tau_p^l \sum_{k=1}^{t} (1-\tau_p^l)^{t-k} \left(a^l \boldsymbol{x}^{l-1}[k] - b^l \boldsymbol{s}^l[k]\right). \tag{9}$$

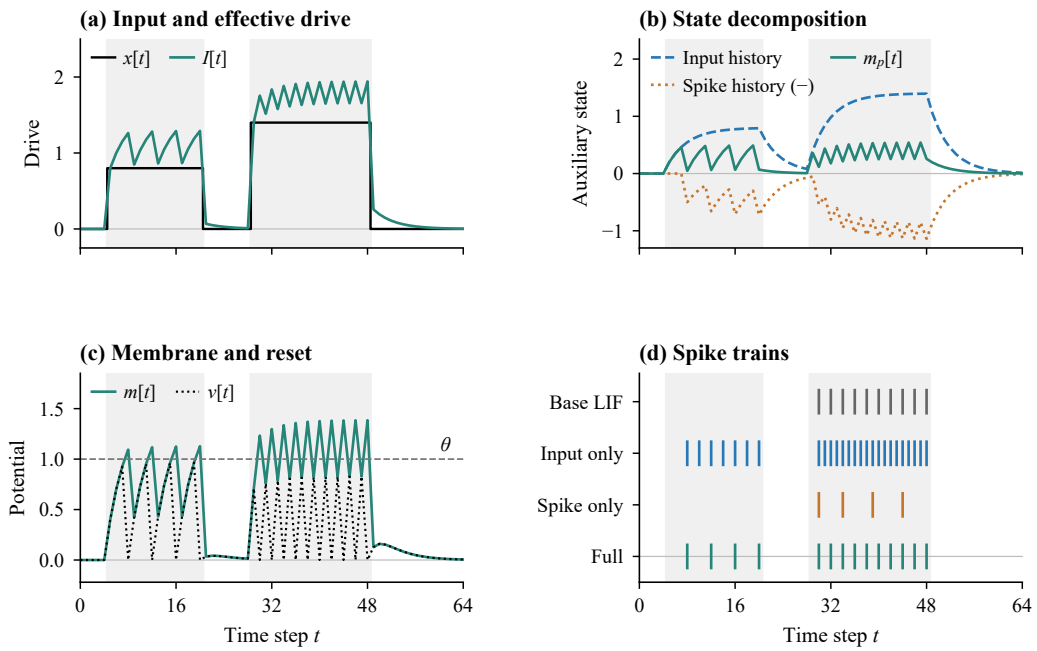


Figure 2: Single-neuron feedback dynamics. (a) External and effective input. (b) Input history and negative spike history, both computed along the full model's trajectory. (c) Pre-reset and post-reset membrane potentials; the dashed line marks the threshold. (d) Four feedback-path variants. Shading marks input presentation. The panels show a 64-step scalar simulation with fixed parameters. Full settings and sensitivity checks are in Appendix.

Thus the auxiliary state is exactly the difference between two exponentially weighted histories. Input history supplies a persistent drive; for $b^l > 0$, an emitted spike decreases the next auxiliary state by $\tau_p^l b^l$ when the preceding state and current input are held fixed. Its downstream influence also depends on the resulting changes in later spikes. The retention factor $(1 - \tau_p^l)$ sets the temporal window, whereas a^l and b^l set the strength and balance of the two contributions.

Setting $b^l = 0$ isolates input memory; setting $a^l = 0$ isolates spike feedback. Removing the auxiliary state recovers the underlying neuron. These ablations distinguish temporal smoothing from output-dependent feedback. A scalar multiplying the entire auxiliary input scales both contributions together; independent a^l, b^l additionally allow their ratio to change.

If $\|\boldsymbol{x}^{l-1}[t]\|_\infty \le X$ and spikes are binary, then

$$\|\boldsymbol{m}_p^l[t]\|_\infty \le \left[1 - (1 - \tau_p^l)^t\right](a^l X + b^l) \le a^l X + b^l. \tag{10}$$

For finite fixed coefficients and bounded inputs, this bound applies throughout the sequence. Exponential parameterization spans positive coefficient values; their evolution and temporal Jacobian products determine the training dynamics.

Single-neuron illustration. Figure 2 separates the forward effects of input and output history. During the sustained subthreshold input $x[t] = 0.8$, base LIF emits no spikes, whereas input-only memory and full feedback emit seven and four, respectively. During the stronger input, their counts are 10, 19 and 10. Thus input history supplies additional drive, while spike feedback moderates the additional firing induced by memory. The auxiliary state also persists across membrane resets. These fixed-parameter trajectories illustrate input amplification, feedback and state retention; Section 5.4 evaluates coefficient adaptation in trained networks.

Table 1: CIFAR-10 accuracy (%). Base and Self-pred denote the base neuron and its original self-prediction variant. Bold marks the higher accuracy within each pair; ties are unbolded.

Network	T	IF		LIF		PLIF		CLIF	
		Base	Self-pred	Base	Self-pred	Base	Self-pred	Base	Self-pred
Spiking CIFAR10Net	4	**94.33**	94.24	93.54	**93.84**	93.85	**94.39**	94.20	**94.70**
	8	94.21	**94.84**	93.60	**94.85**	94.05	**94.85**	94.23	**94.38**
SEW ResNet18	4	**94.53**	94.20	94.99	**95.04**	95.21	**95.25**	94.92	**95.42**
	8	94.50	**94.78**	95.04	**95.56**	95.04	**95.73**	**95.31**	95.26
SEW ResNet34	4	95.49	**95.59**	95.42	**95.86**	95.57	**95.87**	95.54	**96.11**
	8	95.58	**95.73**	94.54	**96.16**	93.49	**96.33**	94.87	**95.99**
Spiking ResNet18	4	94.93	**94.98**	93.43	**95.05**	94.26	**95.05**	95.16	**95.29**
	8	95.16	**95.42**	94.21	**95.41**	94.30	**95.43**	95.52	**95.67**
Spiking ResNet34	4	**94.89**	94.67	91.45	**92.43**	93.16	**93.48**	94.58	**95.04**
	8	94.96	94.96	87.71	**93.82**	88.93	**92.76**	94.57	**94.60**

4.3 Gradient Pathways

We train with surrogate gradients (Neftci et al., 2019), replacing the spike derivative during back-propagation while retaining the hard threshold in the forward pass. The spike is detached in the reset operation but retains its surrogate gradient in the auxiliary-state update in Eq. equation 7. At the level of individual computational-graph edges, the auxiliary update has derivatives

$$\left.\frac{\partial \boldsymbol{m}_p^l[t]}{\partial \boldsymbol{x}^{l-1}[t]}\right|_{\boldsymbol{s}^l[t]} = \tau_p^l a^l \mathbf{Id}, \quad \frac{\partial \boldsymbol{m}_p^l[t]}{\partial \boldsymbol{s}^l[t]} = -\tau_p^l b^l \mathbf{Id}, \quad \left.\frac{\partial \boldsymbol{m}_p^l[t]}{\partial \boldsymbol{m}_p^l[t-1]}\right|_{\boldsymbol{s}^l[t]} = (1-\tau_p^l)\mathbf{Id}. \tag{11}$$

Here $\mathbf{Id}$ denotes the identity matrix. The first edge provides a path from input to auxiliary state that bypasses the current spike nonlinearity. The second transmits an additional surrogate-gradient contribution through the emitted spike. The total state derivative combines explicit retention with the dependence of the spike on the membrane. The full local Jacobian and the update algorithm are given in Appendix.

Temporal credit assignment depends on products of these local Jacobians and on the surrogate derivative along each trajectory. Section 5.4 and Appendix evaluate the input and feedback paths, including detachment of the feedback spike.

Biological motivation. As motivated in the introduction, dendritic modulation and neuronal self-feedback inspire the auxiliary state. The input–output history provides a local drive for subsequent membrane integration. The biological interpretation is discussed in Appendix.

5 Experimental Results

We evaluate the reference feedback rules on CIFAR-10, ImageNet-100, ImageNet-1K, Sequential CIFAR-10, and reinforcement learning tasks. CIFAR-10 ablations in Section 5.4 separately assess learnable coefficients; implementation details and additional ablations are provided in Appendix.

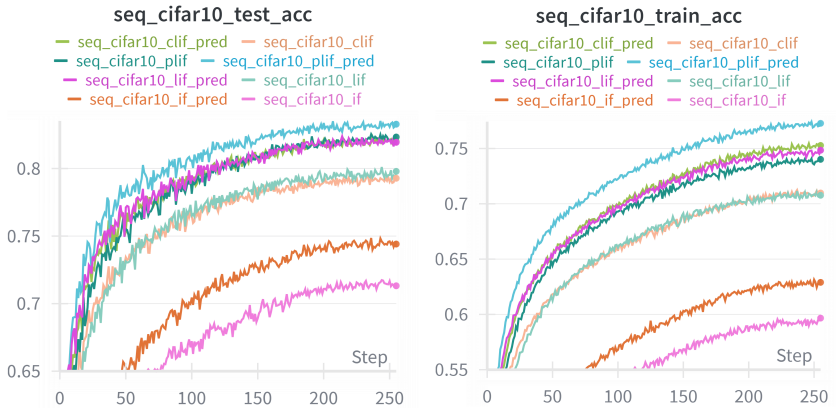
5.1 Evaluation on Classification Tasks

We first validate the effectiveness of our proposed method on the CIFAR-10 dataset. The experiments employ five network architectures, including Spiking CIFAR10Net, SEW ResNet-18, SEW ResNet-34, Spiking ResNet-18, and Spiking ResNet-34; and four spiking neuron models, including IF, LIF, PLIF, and CLIF, summarized in Section 3 with full equations in Appendix, trained under time-steps $T = 4$ and $T = 8$.

Architecture adaptations and layer configurations are provided in Appendix.

Table 2: ImageNet accuracy (%) at $T = 4$. Self-pred denotes the original self-prediction variant.

Dataset	Network	Neuron	Base	Self-pred
ImageNet-100	SEW ResNet34	LIF	65.80	69.06↑
		PLIF	68.30	68.94↑
ImageNet-1K	SEW ResNet18	PLIF	64.25	64.45↑



(a) Test accuracy (b) Training accuracy

Figure 3: The training and testing accuracy curves of four spiking neuron models, IF, LIF, PLIF, and CLIF, along with their variants enhanced by self-prediction mechanisms.

The reference adds one learned τ_p^l per spiking layer and one continuous auxiliary state per neuron. The state requires per-step updates and storage during backpropagation through time. These state operations contribute to the computational and memory cost alongside synaptic processing.

The overall ablation results are presented in Table 1. Accuracy increases in 35 of the 40 configurations, with four decreases and one tie; more configurations improve at $T = 8$ than at $T = 4$. For Spiking ResNet-34 with LIF at $T = 8$, accuracy increases from 87.71% to 93.82%, illustrating the variation in gain across configurations.

We further evaluate large-scale classification on ImageNet-100 with SEW ResNet34 and ImageNet-1K with SEW ResNet18. Table 2 reports gains in all three configurations; the ImageNet-1K increase is 0.20 percentage points.

5.2 Evaluation on Sequence Classification Tasks

To assess our method on sequential inputs, we evaluate it on the Sequential CIFAR-10 task. In this setting, each time step feeds the network with one column of the input image, resulting in a total of $T = 32$ time-steps that matches the image width. As shown in Figure 3, we train models using IF, LIF, PLIF, and CLIF neurons for 256 epochs and plot their training and test accuracy curves. Our self-prediction enhancement improves the best test accuracy for all four neuron types in the reported experiments. Detailed best accuracy results are reported in Table 3.

Table 3: Sequential CIFAR-10 best accuracy (%) at $T = 32$. Detached removes the spike gradient in the feedback update; arrows compare with Base.

Neuron	Base	Self-pred (detached)	Self-pred
IF	71.74	69.08↓	74.86↑
LIF	80.11	81.43↑	82.38↑
PLIF	82.54	83.03↑	83.52↑
CLIF	79.55	81.08↑	82.32↑

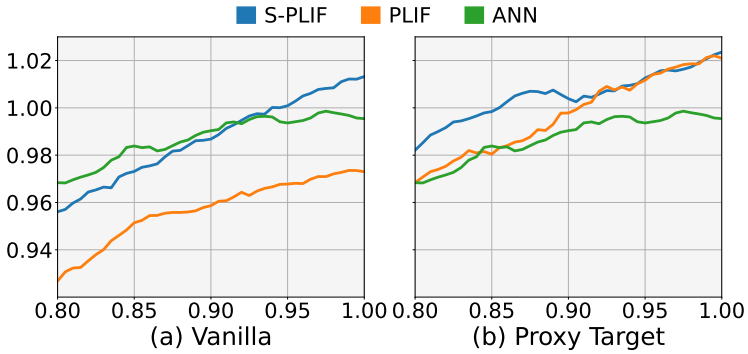


Figure 4: Normalized TD3 learning curves with different spiking neurons, aggregated across all four environments. The performance and training steps are normalized linearly based on ANN performance. Uniform smoothing summarizes the aggregate learning trend; Table 4 reports per-environment return statistics.

Table 4: Maximum average returns across 5 runs (mean $\pm$ one standard deviation). Overall reports the average percentage gain relative to ANN. S-PLIF denotes the original self-prediction variant.

Environment	ANN	Vanilla		PT	
	ReLU	PLIF	S-PLIF	PLIF	S-PLIF
Ant-v4	4932 ± 1287	4912 ± 1009	4981 ± 607	5259 ± 326	5527 ± 151
HalfCheetah-v4	10554 ± 561	9252 ± 520	9828 ± 594	9219 ± 216	9405 ± 597
Hopper-v4	3349 ± 151	3414 ± 122	3462 ± 128	3384 ± 100	3380 ± 105
Walker2d-v4	4050 ± 323	4003 ± 274	4271 ± 485	4445 ± 217	4497 ± 391
Overall	0.00%	-2.99%	0.74%	1.20%	3.28%

Furthermore, we conduct an ablation study on detaching the gradient of the spike signal $\boldsymbol{s}^l[t]$ in the residual $\boldsymbol{x}^{l-1}[t] - b^l \boldsymbol{s}^l[t]$, with $b^l = \tau^l$ for LIF/PLIF and $b^l = 1$ for IF/CLIF, when calculating $\boldsymbol{m}_p^l[t]$. Retaining this gradient improves the reported best accuracy for all four neurons. The paired comparison isolates the effect of retaining this gradient within the full forward recurrence.

5.3 Evaluation on Reinforcement Learning Tasks

We evaluate PLIF and self-prediction enhanced PLIF (S-PLIF) on four MuJoCo continuous-control tasks: Ant-v4, HalfCheetah-v4, Hopper-v4, and Walker2d-v4 (Todorov et al., 2012; Todorov, 2014). We use our TD3 reimplementation (Fujimoto et al., 2018) with vanilla spiking actor networks (Tang et al., 2021) and the proxy target (PT) framework (Xu et al., 2025), following the hyperparameters in Xu et al. (2025). Environment preprocessing, action scaling, and the recorded training settings are given in Appendix.

Figure 4 shows the normalized learning curves across all 4 environments, and Table 4 reports return summaries for each environment. We also report the average performance gain, defined as the average percentage improvement over ANNs, following the metric in Xu et al. (2026). Self-prediction improves mean return in seven of the eight SNN configurations. On Hopper-v4 with PT, mean return changes from 3384 to 3380. S-PLIF exceeds the ANN mean return on Ant-v4, Hopper-v4, and Walker2d-v4 with both actor-training frameworks.

5.4 Ablation of the Feedback Coefficients

Complementing the fixed-parameter dynamics in Fig. 2, we test coefficient adaptation on CIFAR-10 with LIF neurons in SEW ResNet-18 and Spiking ResNet-18. The fixed-feedback reference uses $a^l = 1, b^l = \tau^l = 2$, matching the original LIF implementation; its code-to-notation mapping is given in Appendix. We compare this reference with learning only a^l, only b^l, or both, keeping τ_p^l learnable. All compared runs use 256 epochs, $T = 4$, batch size 64, Adam with initial learning rate

Table 5: Coefficient ablations on CIFAR-10 (LIF, $T = 4$, 256 epochs). Best and Last 5 report maximum official-test accuracy and the mean over epochs 252–256 (%). The fixed reference uses a previously completed run under the shared protocol in Section 5.4.

	SEW ResNet-18		Spiking ResNet-18	
Configuration	Best	Last 5	Best	Last 5
Base LIF	94.90	94.680	93.77	93.450
Fixed feedback reference	95.36	95.170	94.85	94.572
Learn a^l only	95.38	95.250	94.82	94.520
Learn b^l only	95.58	95.288	95.18	94.992
Learn a^l, b^l	94.94	94.700	94.93	94.744

0.001, MSE loss, and the same cosine schedule and augmentation. Table 5 reports both the best official-test accuracy and the final-five-epoch mean.

Learning only b^l improves best accuracy over the fixed-feedback reference by 0.22 and 0.33 percentage points on SEW ResNet-18 and Spiking ResNet-18, respectively. The corresponding final-five-epoch improvements are 0.118 and 0.420 points. Learning only a^l or both coefficients gives architecture-dependent changes. Among the tested coefficient settings, learning only b^l gives the highest Best and Last 5 accuracy on both architectures.

Path ablations in Appendix show that full fixed feedback changes Last 5 accuracy relative to input-only memory by $+0.172$ and -0.226 points on SEW and Spiking ResNet-18, respectively. The relative contribution of spike feedback depends on the architecture. Across the 20 architecture–neuron combinations in Table 10, learning b^l increases best accuracy in 11 and decreases it in nine relative to historical fixed-feedback runs. Protocols and metric definitions are given in Appendix.

6 Conclusion

In this work, we proposed a self-prediction enhanced spiking neuron method, drawing inspiration from the brain's predictive coding mechanisms. By enabling each neuron to generate an internal prediction current based on its own input-output history, our approach offers a simple yet effective way to modulate membrane potential dynamics. The generalized formulation separates input memory and spike feedback while retaining the reference neuron rules as special cases. The auxiliary state introduces additional gradient pathways, and our analysis and simulations clarify how input memory and spike feedback jointly influence neuronal dynamics. Experiments on image classification, sequential classification, and reinforcement learning demonstrate the framework across multiple tasks. LIF ablations show further accuracy gains from learning spike-feedback strength in two architectures. Together, the framework and ablations identify input memory and spike-feedback strength as distinct components for adapting neuronal dynamics.

AI use statement

In this work, we used generative AI tools for writing assistance based on research ideas developed entirely by the authors. We have not used generative AI tools for generating research ideas, developing methodologies, or drawing conclusions, and the remaining disclosure tasks unrelated to writing assistance are not applicable to this work. Additionally, we used generative AI tools for grammar correction, language polishing, and improvements to clarity and readability. We have reviewed all AI-assisted work. We checked LLM-generated research ideas for potential plagiarism through a manual literature survey. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

References

Maximilian Baronig, Romain Ferrand, Silvester Sabathiel, and Robert Legenstein. Advancing spatio-temporal processing through adaptation in spiking neural networks. *Nature Communications*, 16(1):5776, 2025. doi: 10.1038/s41467-025-60878-z.

Guillaume Bellec, Darjan Salaj, Anand Subramoney, Robert Legenstein, and Wolfgang Maass. Long short-term memory and learning-to-learn in networks of spiking neurons. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Johanni Brea, Alexisz Tamás Gaál, Robert Urbanczik, and Walter Senn. Prospective coding by spiking neurons. *PLOS Computational Biology*, 12(6):e1005003, 2016.

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

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.

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

Michael V. DeBole, Brian Taba, Arnon Amir, Filipp Akopyan, Alexander Andreopoulos, William P. Risk, Jeff Kusnitz, Carlos Ortega Otero, Tapan K. Nayak, Rathinakumar Appuswamy, Peter J. Carlson, Andrew S. Cassidy, Pallab Datta, Steven K. Esser, Guillaume J. Garreau, Kevin L. Holland, Scott Lekuch, Michael Mastro, Jeff McKinstry, Carmelo di Nolfo, Brent Paulovicks, Jun Sawada, Kai Schleupen, Benjamin G. Shaw, Jennifer L. Klamo, Myron D. Flickner, John V. Arthur, and Dharmendra S. Modha. TrueNorth: Accelerating from zero to 64 million neurons in 10 years. *Computer*, 52(5):20–29, 2019.

Chaoteng Duan, Jianhao Ding, Shiyan Chen, Zhaofei Yu, and Tiejun Huang. Temporal effective batch normalization in spiking neural networks. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 34377–34390. Curran Associates, Inc., 2022.

Peter Elias. Predictive coding–I. *IRE Transactions on Information Theory*, 1(1):16–24, 1955. doi: 10.1109/TIT.1955.1055126.

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

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

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pp. 1587–1596. PMLR, 2018.

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.

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

Bing Han, Gopalakrishnan Srinivasan, and Kaushik Roy. RMP-SNN: Residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13558–13567, 2020.

Jeff Hawkins and Subutai Ahmad. Why neurons have thousands of synapses, a theory of sequence memory in neocortex. *Frontiers in Neural Circuits*, 10:23, 2016.

Yangfan Hu, Huajin Tang, and Gang Pan. Spiking deep residual networks. *IEEE Transactions on Neural Networks and Learning Systems*, 34(8):5200–5205, 2023.

Yifan Hu, Lei Deng, Yujie Wu, Man Yao, and Guoqi Li. Advancing spiking neural networks toward deep residual learning. *IEEE Transactions on Neural Networks and Learning Systems*, 36(2): 2353–2367, 2025.

Yanping Huang and Rajesh PN Rao. Predictive coding. *Wiley Interdisciplinary Reviews: Cognitive Science*, 2(5):580–593, 2011.

Yulong Huang, Xiaopeng Lin, Hongwei Ren, Haotian Fu, Yue Zhou, Zunchang Liu, Biao Pan, and Bojun Cheng. CLIF: Complementary leaky integrate-and-fire neuron for spiking neural networks. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 19949–19972. PMLR, 2024a.

Zihan Huang, Xinyu Shi, Zecheng Hao, Tong Bu, Jianhao Ding, Zhaofei Yu, and Tiejun Huang. Towards high-performance spiking transformers from ANN to SNN conversion. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pp. 10688–10697, 2024b.

Mengting Lan, Xiaogang Xiong, Zixuan Jiang, and Yunjiang Lou. PC-SNN: Supervised learning with local Hebbian synaptic plasticity based on predictive coding in spiking neural networks. *arXiv preprint arXiv:2211.15386*, 2022.

Ana B Lao-Rodríguez, Karol Przewrocki, David Pérez-González, Artoghrul Alishbayli, Evrim Yilmaz, Manuel S Malmierca, and Bernhard Englitz. Neuronal responses to omitted tones in the auditory brain: A neuronal correlate for predictive coding. *Science Advances*, 9(24):eabq8657, 2023.

Ana B Lao-Rodríguez, Erich Schröger, and Manuel S Malmierca. The sound of silence: Omission responses and how the brain predicts in the absence of sound. *Neuroscience & Biobehavioral Reviews*, 181:106505, 2026. doi: 10.1016/j.neubiorev.2025.106505.

Kwangjun Lee, Shirin Dora, Jorge F Mejias, Sander M Bohte, and Cyriel MA Pennartz. Predictive coding with spiking neurons and feedforward gist signaling. *Frontiers in Computational Neuroscience*, 18:1338280, 2024.

Guoqi Li, Lei Deng, Huajin Tang, Gang Pan, Yonghong Tian, Kaushik Roy, and Wolfgang Maass. Brain-inspired computing: A systematic survey and future trends. *Proceedings of the IEEE*, 112 (6):544–584, 2024. doi: 10.1109/JPROC.2024.3429360.

Yuhang Li, Shikuang Deng, Xin Dong, Ruihao Gong, and Shi Gu. A free lunch from ANN: Towards efficient, accurate spiking neural networks calibration. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pp. 6316–6325. PMLR, 2021.

Ming Liang and Xiaolin Hu. Recurrent convolutional neural network for object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3367–3375, 2015.

Ming Liang, Xiaolin Hu, and Bo Zhang. Convolutional neural networks with intra-layer recurrent connections for scene labeling. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

Qianli Liao and Tomaso Poggio. Bridging the gaps between residual learning, recurrent neural networks and visual cortex. *arXiv preprint arXiv:1604.03640*, 2016.

De Ma, Juncheng Shen, Zonghua Gu, Ming Zhang, Xiaolei Zhu, Xiaoqiang Xu, Qi Xu, Yangjing Shen, and Gang Pan. Darwin: A neuromorphic hardware co-processor based on spiking neural networks. *Journal of Systems Architecture*, 77:43–51, 2017.

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

Timothée Masquelier and Simon J Thorpe. Unsupervised learning of visual features through spike timing dependent plasticity. *PLOS Computational Biology*, 3(2):e31, 2007.

Timothée Masquelier, Rudy Guyonneau, and Simon J Thorpe. Spike timing dependent plasticity finds the start of repeating patterns in continuous spike trains. *PLOS ONE*, 3(1):e1377, 2008.

Timothée Masquelier, Rudy Guyonneau, and Simon J Thorpe. Competitive STDP-based spike pattern learning. *Neural Computation*, 21(5):1259–1276, 2009.

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

Jing Pei, Lei Deng, Sen Song, Mingguo Zhao, Youhui Zhang, Shuang Wu, Guanrui Wang, Zhe Zou, Zhenzhi Wu, Wei He, Feng Chen, Ning Deng, Si Wu, Yu Wang, Yujie Wu, Zheyu Yang, Cheng Ma, Guoqi Li, Wentao Han, Huanglong Li, Huaqiang Wu, Rong Zhao, Yuan Xie, and Luping Shi. Towards artificial general intelligence with hybrid Tianjic chip architecture. *Nature*, 572(7767): 106–111, 2019.

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. ISSN 1662-453X. doi: 10.3389/fnins.2017.00682.

João Sacramento, Rui Ponte Costa, Yoshua Bengio, and Walter Senn. Dendritic cortical microcircuits approximate the backpropagation algorithm. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Iria SanMiguel, Andreas Widmann, Alexandra Bendixen, Nelson Trujillo-Barreto, and Erich Schröger. Hearing silences: human auditory processing relies on preactivation of sound-specific brain activity patterns. *Journal of Neuroscience*, 33(20):8633–8639, 2013.

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

Michael W Spratling. A review of predictive coding algorithms. *Brain and Cognition*, 112:92–97, 2017.

Guangzhi Tang, Neelesh Kumar, Raymond Yoo, and Konstantinos Michmizos. Deep reinforcement learning with population-coded spiking neural network for continuous control. In *Proceedings of the 2020 Conference on Robot Learning*, volume 155, pp. 2016–2029. PMLR, 2021.

Emanuel Todorov. Convex and analytically-invertible dynamics with contacts and constraints: Theory and implementation in MuJoCo. In *2014 IEEE International Conference on Robotics and Automation*, pp. 6054–6061. IEEE, 2014.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

Robert Urbanczik and Walter Senn. Learning by the dendritic prediction of somatic spiking. *Neuron*, 81(3):521–528, 2014.

Lihao Wang and Zhaofei Yu. Autaptic synaptic circuit enhances spatio-temporal predictive learning of spiking neural networks. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 52083–52098, 2024.

Siqi Wang, Tee Hiang Cheng, and Meng-Hiot Lim. LTMD: learning improvement of spiking neural networks with learnable thresholding neurons and moderate dropout. In *Advances in Neural Information Processing Systems*, volume 35, pp. 28350–28362, 2022.

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.

Zijie Xu, Tong Bu, Zecheng Hao, Jianhao Ding, and Zhaofei Yu. Proxy target: Bridging the gap between discrete spiking neural networks and continuous control. In *Advances in Neural Information Processing Systems*, volume 38, pp. 159158–159184, 2025.

Zijie Xu, Xinyu Shi, Yiting Dong, Zihan Huang, and Zhaofei Yu. CaRe-BN: Precise moving statistics for stabilizing spiking neural networks in reinforcement learning. In *International Conference on Learning Representations*, 2026.

Man Yao, Jiakui Hu, Zhaokun Zhou, Li Yuan, Yonghong Tian, Bo Xu, and Guoqi Li. Spike-driven transformer. In *Advances in Neural Information Processing Systems*, volume 36, pp. 64043–64058, 2023.

Xingting Yao, Fanrong Li, Zitao Mo, and Jian Cheng. GLIF: A unified gated leaky integrate-and-fire neuron for spiking neural networks. In *Advances in Neural Information Processing Systems*, volume 35, pp. 32160–32171, 2022.

Amit Yaron, Tomoyo Shiramatsu-Isoguchi, Felix B Kern, Kenichi Ohki, Hirokazu Takahashi, and Zenas C Chao. Auditory cortex neurons that encode negative prediction errors respond to omissions of sounds in a predictable sequence. *PLOS Biology*, 23(6):e3003242, 2025.

Bojian Yin, Federico Corradi, and Sander M Bohté. Effective and efficient computation with multiple-timescale spiking recurrent neural networks. In *International Conference on Neuromorphic Systems 2020*, pp. 1–8, 2020.

Luping Yin, Rui Zheng, Wei Ke, Quansheng He, Yi Zhang, Junlong Li, Bo Wang, Zhen Mi, Yuesheng Long, Malte J. Rasch, Tianfu Li, Guoming Luan, and Yousheng Shu. Autapses enhance bursting and coincidence detection in neocortical pyramidal cells. *Nature Communications*, 9(1): 4890, 2018.

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

Chenlin Zhou, Han Zhang, Zhaokun Zhou, Liutao Yu, Liwei Huang, Xiaopeng Fan, Li Yuan, Zhengyu Ma, Huihui Zhou, and Yonghong Tian. QKFormer: Hierarchical spiking transformer using Q-K attention. In *Advances in Neural Information Processing Systems*, volume 37, pp. 13074–13098, 2024.

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

A RESULTS ON CIFAR10 DATASET

A.1 RESULTS ON CIFAR10NET

This Cifar10Net architecture contains six convolutional blocks, each with a 3×3 convolution, stride 1, padding 1, 256 output channels, batch normalization, and a spiking neuron. The first convolution accepts three input channels, and all subsequent convolutions have 256 input channels. A 2×2 max-pooling layer follows each group of three blocks. The flattened features pass through dropout with probability 0.5, Linear($256 \times 8 \times 8$, $128 \times 4 \times 4$) with a spiking neuron, and Linear($128 \times 4 \times 4$, 100) with a spiking neuron. An AvgPool1d(10, 10) readout averages each group of ten output neurons into one class score; the scores are summed over time. Training and testing curves are shown in Figure 5.

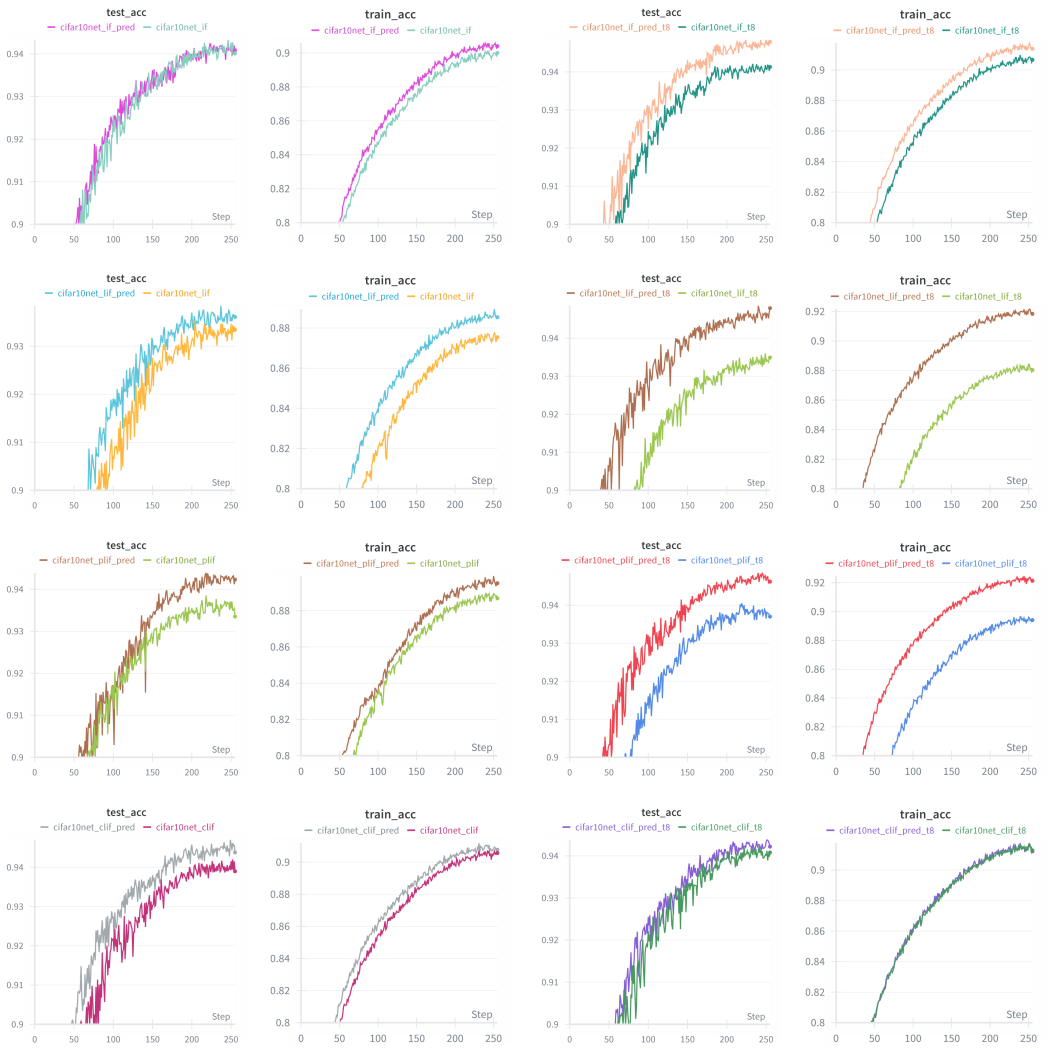


Figure 5: The training and testing accuracy curves of the CIFAR10Net trained with IF,LIF,PLIF and CLIF neurons along with their variants enhanced by self-prediction mechanisms at time-steps T=4 and T=8.

A.2 RESULTS ON SEW RESNET

For the SEW ResNet families which is originally designed for ImageNet, when adapting them to CIFAR-10, we modify the first convolutional layer parameters, changing kernel size, stride, and padding from 7, 2, 3 to 3, 1, 1, respectively, and replace the initial max pooling layer with an identity mapping. Figure 6 shows the training and testing accuracy curve on SEW ResNet18. Figure 7 shows the training and testing accuracy curve on SEW ResNet34.

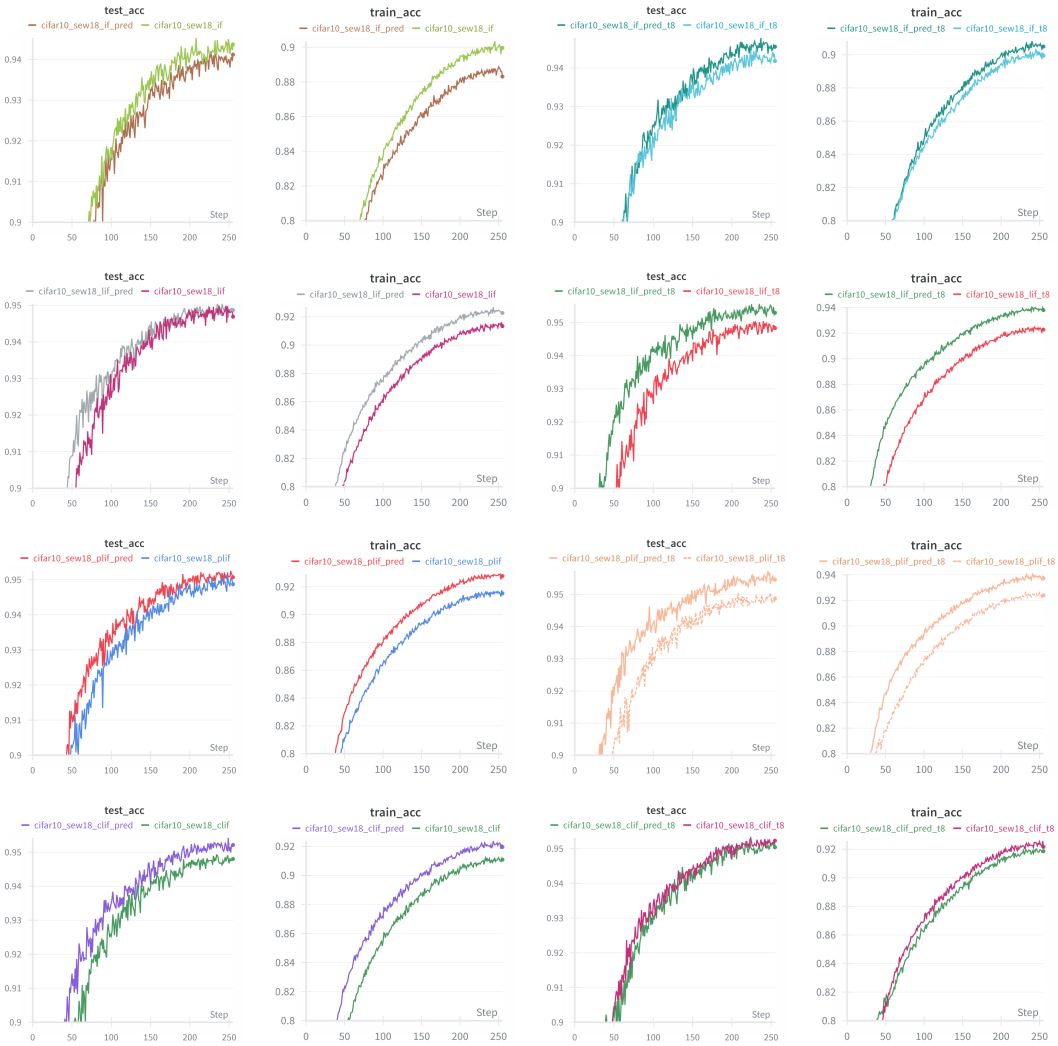


Figure 6: The training and testing accuracy curves of the SEW ResNet18 trained on CIFAR10 dataset with IF,LIF,PLIF and CLIF neurons along with their variants enhanced by self-prediction mechanisms at time-steps T=4 and T=8.

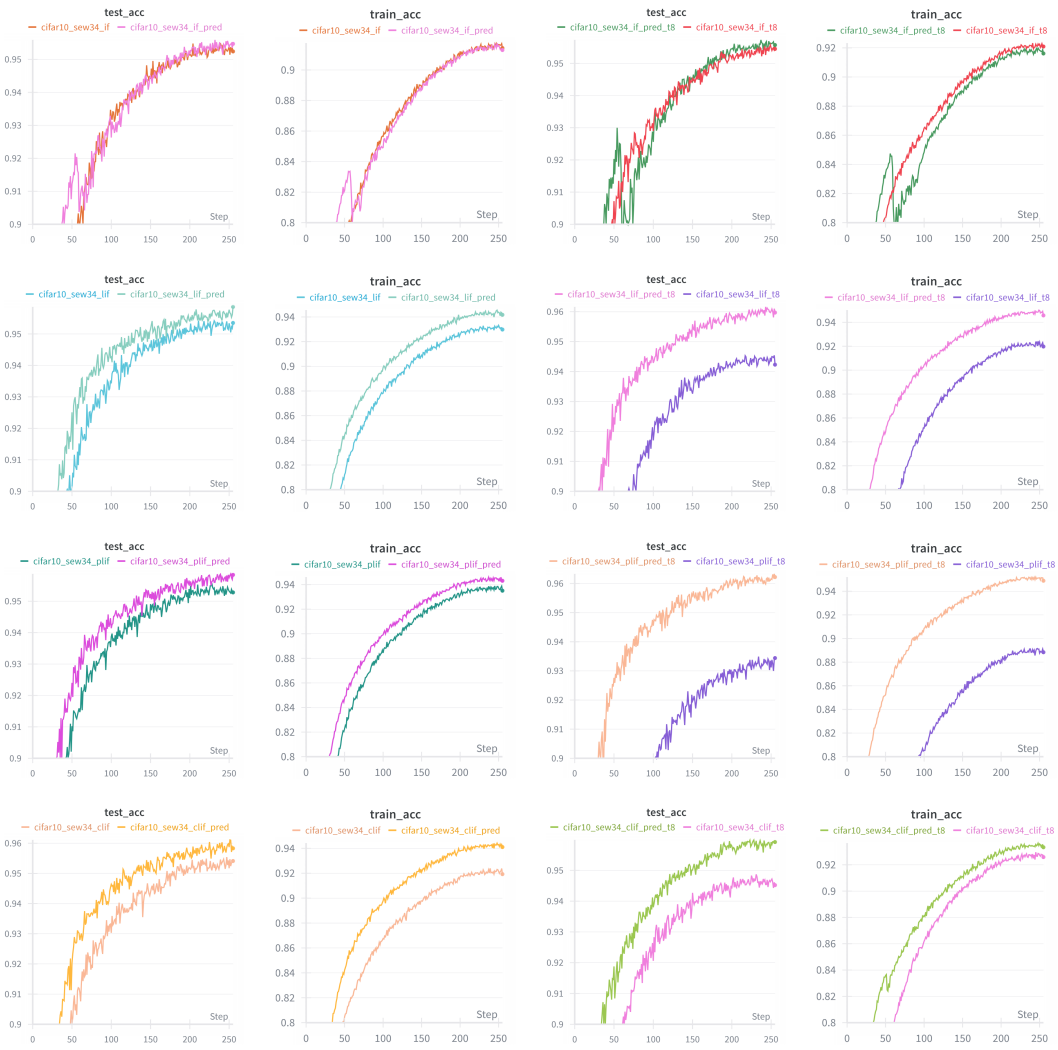


Figure 7: The training and testing accuracy curves of the SEW ResNet34 trained on CIFAR10 dataset with IF,LIF,PLIF and CLIF neurons along with their variants enhanced by self-prediction mechanisms at time-steps T=4 and T=8.

A.3 RESULTS ON SPIKING RESNET

For the Spiking ResNet families which is originally designed for ImageNet, when adapting them to CIFAR-10, we also modify the first convolutional layer parameters, changing kernel size, stride, and padding from 7, 2, 3 to 3, 1, 1, respectively, and replace the initial max pooling layer with an identity mapping. Figure 8 shows the training and testing accuracy curve on Spiking ResNet18. Figure 9 shows the training and testing accuracy curve on Spiking ResNet34.

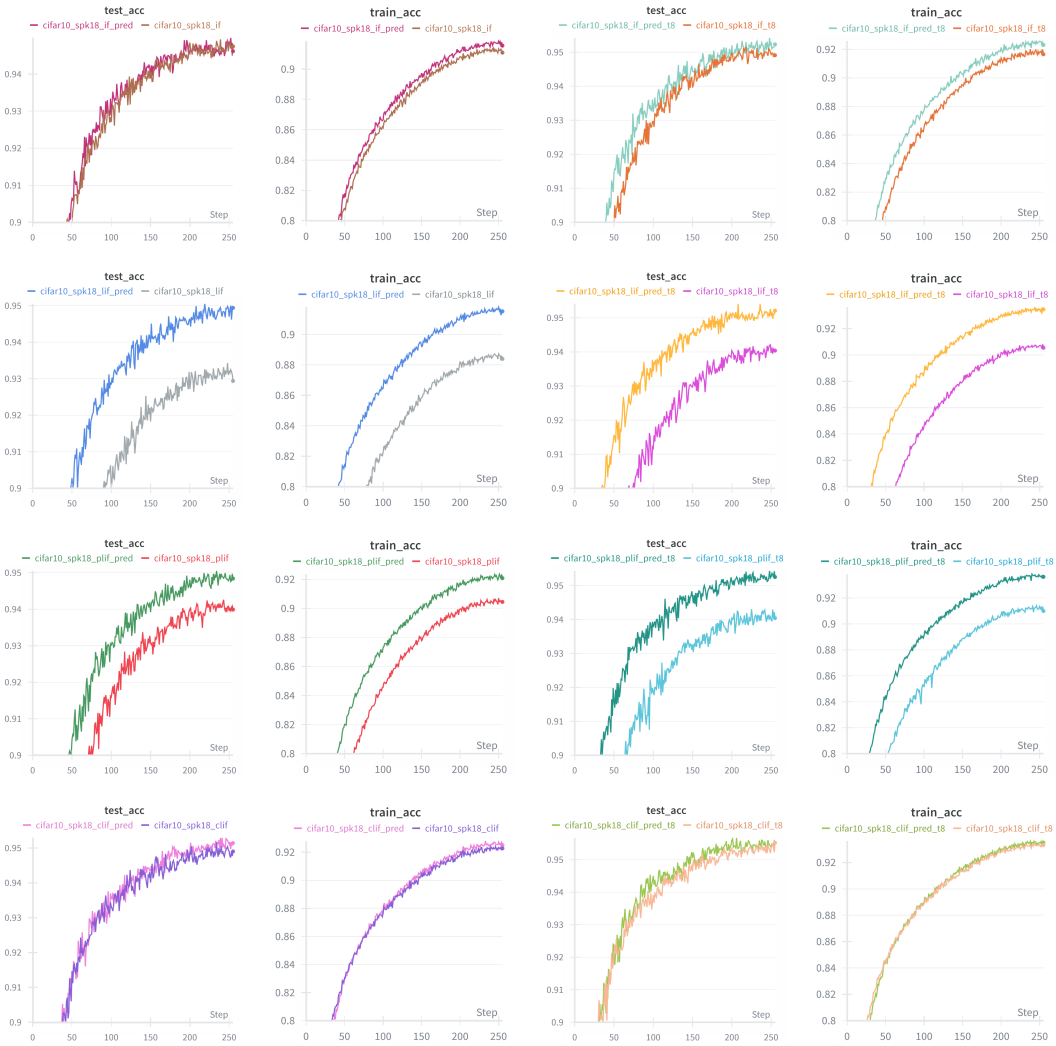


Figure 8: The training and testing accuracy curves of the Spiking ResNet18 trained on CIFAR10 dataset with IF,LIF,PLIF and CLIF neurons along with their variants enhanced by self-prediction mechanisms at time-steps T=4 and T=8.

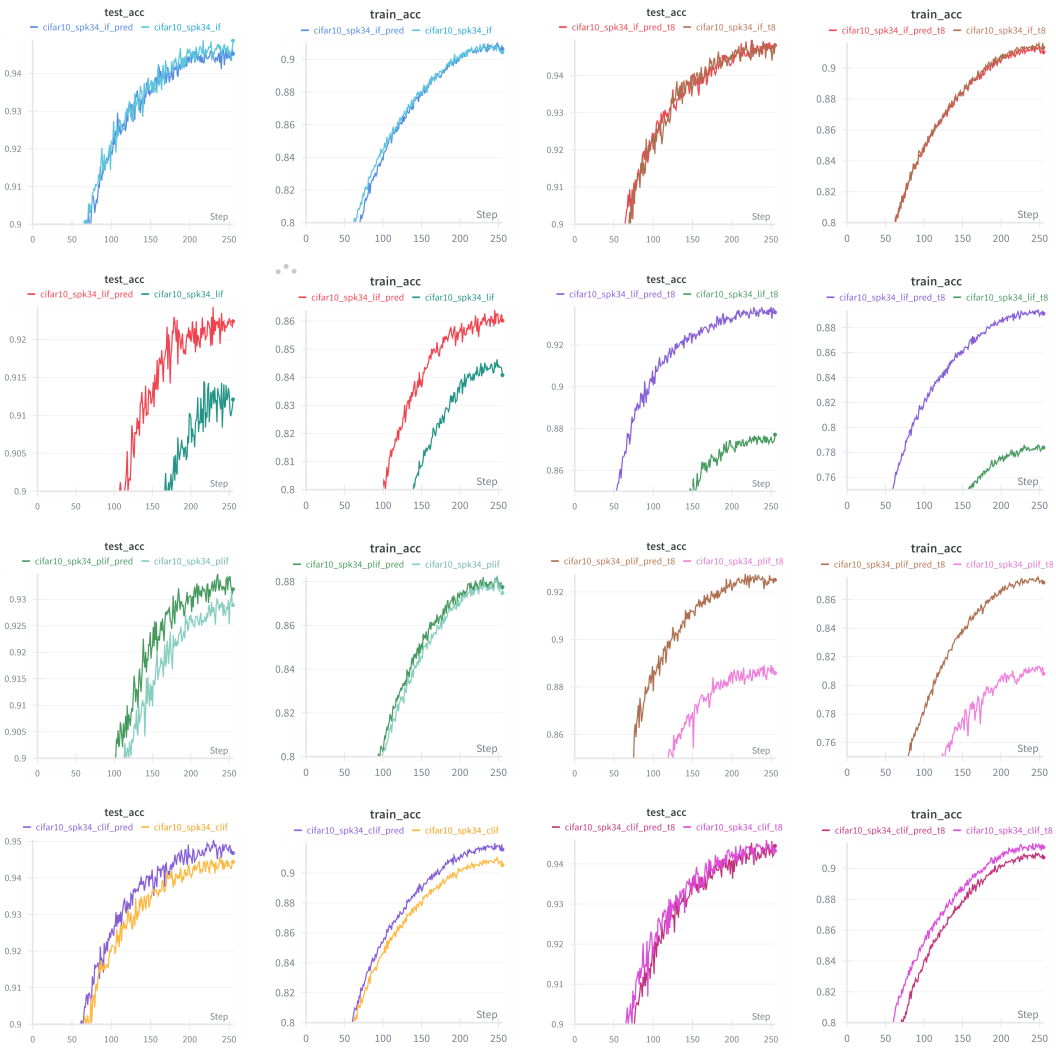


Figure 9: The training and testing accuracy curves of the Spiking ResNet34 trained on CIFAR10 dataset with IF,LIF,PLIF and CLIF neurons along with their variants enhanced by self-prediction mechanisms at time-steps T=4 and T=8.

B RESULTS ON IMAGENET100 AND IMAGENET1K DATASET

Due to computational resource constraints, we evaluated the performance gains of the self-prediction enhancement method using the SEW ResNet-34 model on ImageNet-100 with both LIF and PLIF neurons. Additionally, we tested its effectiveness with PLIF neurons on the full ImageNet dataset using the SEW ResNet-18 model. The corresponding training and testing curves are shown in the Figure 10.

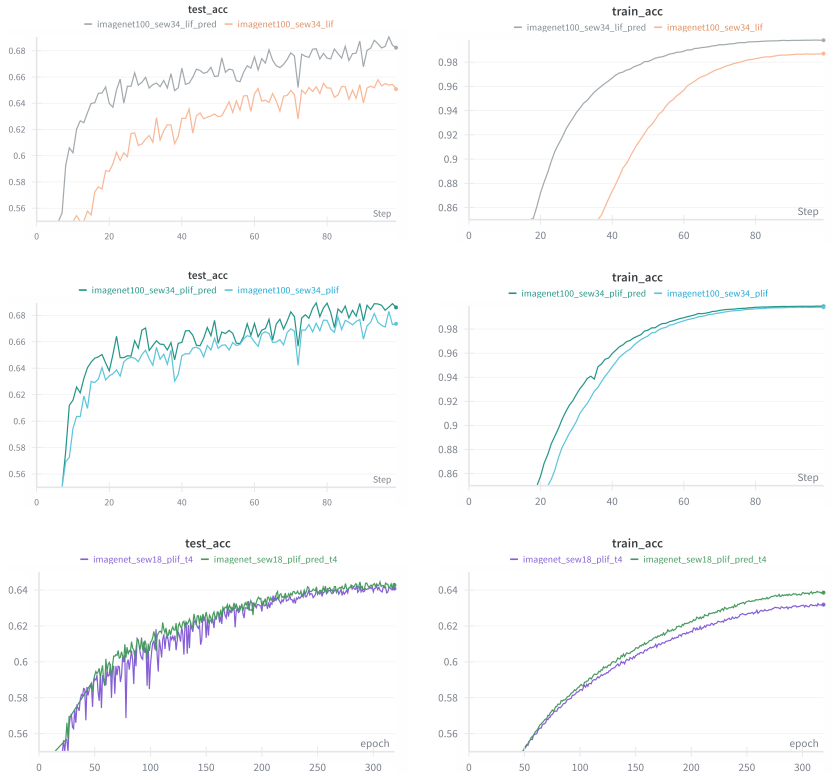


Figure 10: Training and testing accuracy curves on ImageNet100 and ImageNet dataset.

C Results on Sequential CIFAR10 Classification Tasks

For Sequential CIFAR10 Classification Tasks, we use our CIFAR10NetSeq architecture , which processes input images in a sequential manner by treating the spatial width dimension as time steps. It consists of the following components in sequence: two macro stages, each containing three convolutional blocks followed by an average pooling layer. Each convolutional block comprises Conv1d(inchannels, channels=128, kernelsize=3, padding=1, bias=False) to BatchNorm1d(128) to spiking neuron. The first block uses 3 input channels (RGB), while all subsequent blocks use 128 channels. After every three convolutional blocks, an AvgPool1d(2) halves the spatial height within each input column, from 32 to 16 and then 8; the temporal length remains 32 steps. Following the convolutional stack, the feature sequence is flattened and passed through two fully connected layers: Linear(128×8, 256) with a spiking neuron, then Linear(256, 10) without a neuron. Figure 11 shows the training and testing accuracy curve.

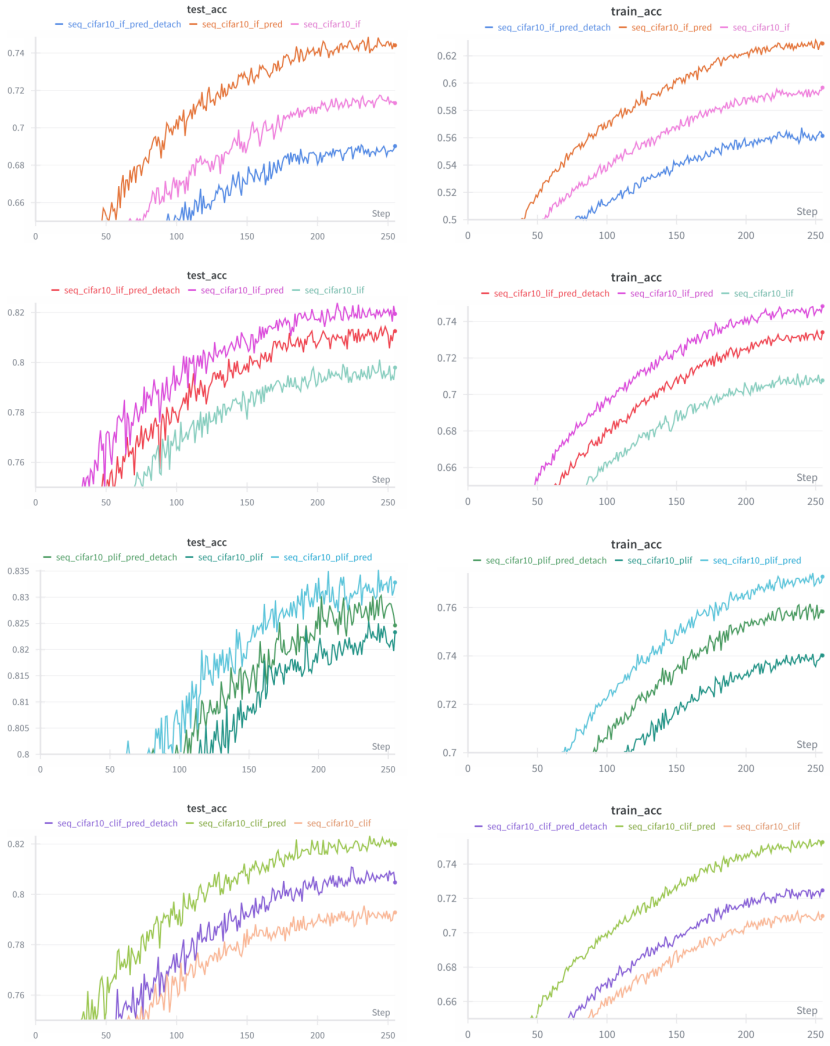


Figure 11: Training and testing accuracy curves on sequential CIFAR-10 using IF, LIF, PLIF, and CLIF neurons, including their self-prediction enhanced variants and an ablation on spike detachment in the computation of $\boldsymbol{m}_p^l[t]$.

D EXPERIMENT DETAILS OF REINFORCEMENT LEARNING TASKS

D.1 SPECIFIC PARAMETERS FOR THE PROXY TARGET FRAMEWORK

Table 6 lists the recorded proxy-network settings, with wider hidden layers than the spiking actor.

D.2 SPIKING ACTOR NETWORK PARAMETERS

All hyperparameters of the spiking actor network are summarized in Table 7, following the same configuration as in a wide range of prior studies (Tang et al., 2021).

Table 6: Proxy network hyperparameters.

Parameter	LIF
Proxy network architecture	$(512, 512)$
Proxy network activation	ReLU
Proxy network learning rate	$3 \cdot 10^{-3}$
Proxy network optimizer	Adam
Proxy update iterations K	1
Proxy update batch size N	256

Table 7: Spiking actor hyperparameters.

Parameter	Value
Encoder population per dimension N_{in}	10
Encoder threshold V_E	0.999
Network hidden units	$(256, 256)$
Decoder population per dimension N_{out}	10
Surrogate window size ω	0.5

D.3 RL ALGORITHM PARAMETERS

We conduct our experiments based on the TD3 algorithm (Fujimoto et al., 2018), using the hyper-parameters listed in Table 8.

Table 8: Hyper-parameters of the implemented TD3 algorithm (Fujimoto et al., 2018)

Parameter	Value	Parameter	Value
Actor learning rate	$3 \cdot 10^{-4}$	Discount factor γ	0.99
Actor regularization	None	Iterations per time step	1.0
Critic learning rate	$3 \cdot 10^{-4}$	Reward scaling	1.0
Critic regularization	None	Gradient clipping	None
Critic architecture	$(256, 256)$	Replay buffer size	10^6
Critic activation	ReLU	Exploration noise $\mathcal{N}(0, \sigma)$	$\mathcal{N}(0, 0.1)$
Optimizer	Adam	Actor update interval d	2
Target update rate τ	$5 \cdot 10^{-3}$	Target policy noise $\mathcal{N}(0, \tilde{\sigma})$	$\mathcal{N}(0, 0.2)$
Batch size N	256	Target policy noise clip c	0.5

D.4 EXPERIMENT ENVIRONMENTS

All environments were used with their default configurations without any modifications. Notably, since the state vector can take values in $-\infty$ to ∞, it is normalized to the range $(-1, 1)$ using a tanh function. Similarly, because actions are bounded by predefined minimum and maximum limits, the actor network's output is first passed through a tanh function to constrain it to $(-1, 1)$, and then linearly scaled to the actual action range $(\text{Min action}, \text{Max action})$.

Return aggregation. For each method, Overall in Table 4 is $\frac{100}{4} \sum_e (R_{\text{method},e}/R_{\text{ANN},e} - 1)$, where R is the reported mean return and e indexes the four environments. Each environment receives equal weight. Figure 4 summarizes the normalized and smoothed aggregate trajectory.

E Feedback Dynamics and Implementation Details

E.1 Implementation Conventions

The membrane time constant is τ^l in Eq. equation 2. Its reciprocal $c_m^l = 1/\tau^l$ is the update coefficient, named `tau` in the original LIF code. The coded residual $\boldsymbol{x}^{l-1}[t] - \boldsymbol{s}^l[t]/c_m^l = \boldsymbol{x}^{l-1}[t] - \tau^l \boldsymbol{s}^l[t]$ therefore gives $b^l = \tau^l$, consistent with Eq. equation 8; code `tau`$= 0.5$ gives $\tau^l = b^l = 2$.

All reference implementations use $a^l = 1$. PLIF uses $c_m^l = \sigma(w^l)$ and $b^l = 1/\sigma(w^l) = \tau^l$; IF and CLIF use $b^l = 1$, independent of the membrane time constant. LIF ablations start from $a_0^l = 1, b_0^l = 2$.

For the coefficient ablations, $a^l = \exp(\phi_a^l)$ and $b^l = 2\exp(\phi_b^l)$ with zero initialization. The unselected coefficient is fixed, and τ_p^l remains learned through its existing sigmoid parameterization. The membrane update, threshold, reset, surrogate, optimizer and augmentation are held fixed across the LIF comparisons. Learning the feedback coefficient preserves the same temporal state variables. In the broader PLIF experiments, it also removes the dependence of the feedback coefficient on the learned membrane parameter, while membrane integration remains learnable; those runs therefore test decoupling as well as coefficient adaptation.

E.2 Single-Neuron Dynamics

Simulation protocol. We evaluate the feedback recurrence in Eqs. equation 5–equation 7, coupled to the LIF membrane, spike and hard-reset updates in Eqs. equation 2–equation 4, with zero initial states, $\tau = 2$, $\theta = 1$, and $v_{\text{reset}} = 0$, omitting layer indices. Each step uses $m_p[t-1]$ for membrane integration, generates $s[t] = \mathbb{1}_{m[t] \geq \theta}$, resets the membrane, and then updates $m_p[t]$. The reference in Fig. 2 has fixed $a = 1, b = 2, \tau_p = 0.25$. The sensitivity grid also includes the implementation's initialization value $\tau_p = 0.5$. The other variants use $(a, b) = (0, 0)$ (base LIF), $(1, 0)$ (input only), and $(0, 2)$ (spike only). Input is 0.8 at steps 5–20 and 1.4 at steps 29–48, and zero otherwise, over 64 steps. The state is not reset between input segments. Computation uses float64, with no randomness or threshold tolerance; connecting lines visualize discrete samples.

Exact history decomposition. For zero initial state, define two analysis quantities along the full model's trajectory:

$$h_x[t] = \tau_p a \sum_{k=1}^{t} (1 - \tau_p)^{t-k} x[k], \tag{12}$$

$$h_s[t] = -\tau_p b \sum_{k=1}^{t} (1 - \tau_p)^{t-k} s[k]. \tag{13}$$

Then $m_p[t] = h_x[t] + h_s[t]$ exactly. Both quantities are analytical components of the full-model trajectory; the spike-only variant follows its own recurrence. Direct recurrence, convolution and history-sum evaluations agree to within 5×10^{-16} in the saved trajectories.

The input-period spike counts for base, input-only, spike-only and full models are respectively $(0, 7, 0, 4)$ during the weak segment and $(10, 19, 4, 10)$ during the strong segment. Input-only and full feedback both first fire at $t = 8$. At that step $v[8] = 0$ but $m_p[8] = 0.046875$, illustrating history retention across membrane reset. All four models have zero post-input spikes in both observation windows of the main example.

One-step causal check. We retain the naturally generated spike and hard reset at $t = 8$, but omit that spike's negative contribution only in the auxiliary update. Current $I[8], m[8], v[8], s[8]$ remain unchanged, while the intervened state increases by $\Delta m_p[8] = \tau_p b = 0.5$. At the next step, $\Delta I[9] = 0.5$ and $\Delta m[9] = 0.25$. These differences compare the intervened and natural trajectories at matched time steps. Subsequent effects depend on changes in later spikes.

Input amplitude and memory-rate sensitivity. Figure 12 varies input amplitude and memory rate with reference feedback strength $b = \tau = 2$. The grid comprises nine amplitudes from 0.4

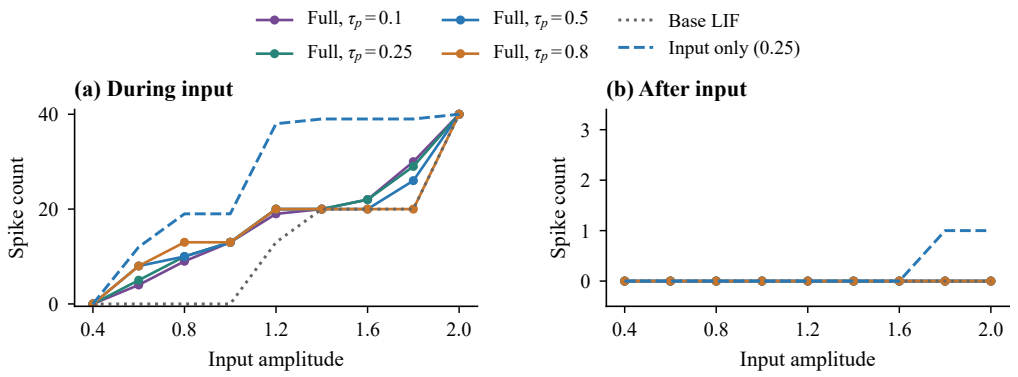


Figure 12: Input-amplitude and memory-rate sensitivity with reference feedback strength $b = \tau = 2$. Panels count spikes (a) during input and (b) after withdrawal. All full-feedback rates are shown. The plotted input-only reference uses $\tau_p = 0.25$; base LIF is independent of this rate. The matched-rate counts in the text use the complete numerical grid. Lines connect the sampled settings.

to 2.0 in steps of 0.2, four rates $\tau_p \in \{0.1, 0.25, 0.5, 0.8\}$, and four variants: 144 deterministic sequences. Each starts from zero, contains 40 input steps (5–44), and ends with 56 zero-input steps. Relative to input-only memory at the same rate, full feedback reduces input-period spike counts in 28/36 settings; the other settings tie. All 36 full-feedback settings have zero post-input spikes. At amplitude 2.0, some full-feedback settings fire at every input step.

Input amplification and memory. The scalar simulations examine forward dynamics over 64 or 100 steps with prescribed coefficients. Weak-input firing includes input amplification: before firing under constant input, $m_p \to ax$ and $I \to (1 + a)x$. Task-level evaluation uses the trained $T = 4$ networks reported in Section 5.4 and Appendix.

Biological interpretation. Predictive coding motivates the local input–output residual. Dendritic modulation (Sacramento et al., 2018) and neuronal autapses (Yin et al., 2018) motivate a separate modulatory state and feedback of the neuron's own output. The biological motivation is neuron-local feedback: pyramidal-cell autapses in the cited study are excitatory, whereas $b^l > 0$ implements subtractive spike feedback here. Self-prediction refers to the resulting history-dependent drive for subsequent integration. The auxiliary current can amplify weak inputs sufficiently to trigger spikes.

E.3 Unrolled State and Conditional Boundedness

Repeated substitution of Eq. equation 7 gives Eq. equation 9. For zero initial auxiliary state, $a^l, b^l \geq 0$, binary spikes, and $\|\boldsymbol{x}^{l-1}[k]\|_\infty \leq X$, the triangle inequality yields

$$\|\boldsymbol{m}_p^l[t]\|_\infty \leq \tau_p^l \sum_{k=1}^{t} (1 - \tau_p^l)^{t-k}(a^l X + b^l) \tag{14}$$

$$= \left[1 - (1 - \tau_p^l)^t\right](a^l X + b^l). \tag{15}$$

For the hard-reset LIF example with $v_{\text{reset}} = 0$ and $0 < 1/\tau^l < 1$, the bounded auxiliary drive and bounded input also bound the membrane magnitude by its leaky recurrence. This per-sequence bound assumes finite fixed coefficients; training dynamics additionally depend on parameter evolution and products of surrogate-gradient Jacobians.

E.4 Full Local Surrogate Jacobian

For a sigmoid surrogate, the spike derivative in backpropagation is replaced by

$$\frac{\partial \boldsymbol{s}^l[t]}{\partial \boldsymbol{m}^l[t]} \approx \sigma'(\boldsymbol{m}^l[t] - \theta^l), \qquad \sigma(x) = \frac{1}{1 + e^{-kx}}, \quad k > 0. \tag{16}$$

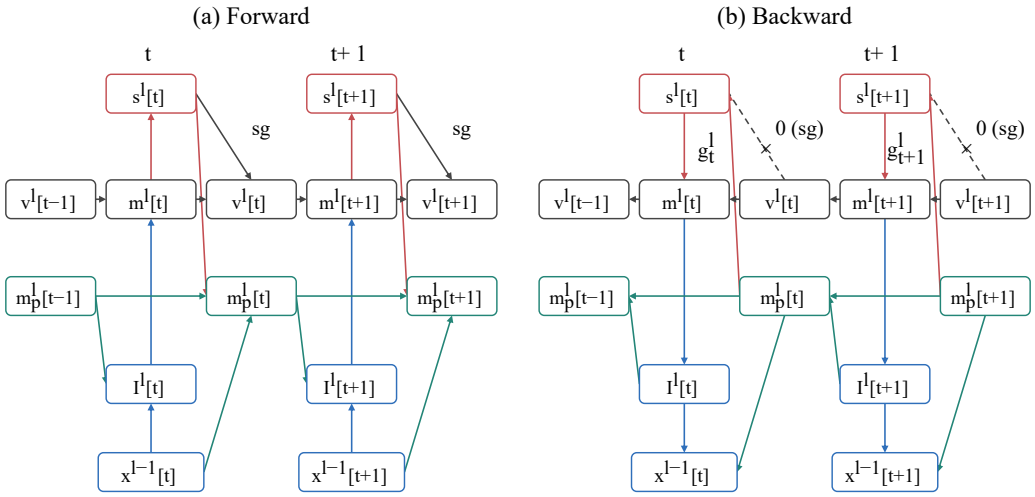


Figure 13: Forward (a, left) and backward (b, right) dependencies over two time steps. Blue, teal and red indicate membrane input, auxiliary memory/input history and spike-related paths, respectively. Backward arrows reverse forward dependencies. The dashed reset branch has zero gradient under sg; the spike-dependent auxiliary update retains the surrogate gradient g_k^l evaluated at $m^l[k] - \theta^l$. Crossings without junction dots add no dependencies. Edge derivatives and the full state Jacobian are given in Eqs. equation 11 and equation 17.

Here k sets the sigmoid steepness; the hard threshold remains unchanged in the forward pass. Figure 13 shows the forward and backward dependencies. The analysis below is expressed in terms of a generic surrogate derivative.

Consider one neuron, suppress the layer index, and hold its current input fixed. Let g_t be the surrogate derivative evaluated at $m[t] - \theta$, and let $s[t]$ in the reset multiplier be treated as constant by the detached-reset convention. For the state $(v[t], m_p[t])$, the local surrogate Jacobian with respect to $(v[t-1], m_p[t-1])$ is

$$\mathbf{J}_t = \begin{pmatrix} (1-s[t])(1-1/\tau) & (1-s[t])/\tau \\ -\tau_p b g_t (1-1/\tau) & 1-\tau_p-\tau_p b g_t/\tau \end{pmatrix}. \tag{17}$$

The input derivative is

$$\frac{\partial(v[t], m_p[t])}{\partial x[t]} = \begin{pmatrix} (1-s[t])/\tau \\ \tau_p a - \tau_p b g_t/\tau \end{pmatrix}. \tag{18}$$

These derivatives describe the surrogate computational graph used for training. Detaching the spike only in the auxiliary update removes the terms containing bg_t from the second row of these expressions, while leaving the forward update unchanged. Across multiple steps, products of the local Jacobians determine gradient propagation.

Here sg leaves the forward value unchanged and stops the gradient. The threshold operation uses the existing surrogate derivative in backpropagation. Each independent input sequence starts with reset states. Algorithm 1 advances the neuron states in causal time order.

Algorithm 1 Generalized local feedback: hard-reset LIF example

Require: Sequence $\{\boldsymbol{x}^{l-1}[t]\}_{t=1}^{T}$; parameters $\tau^l, \tau_p^l, a^l, b^l, \theta^l, v_{\text{reset}}$.

Initialize $\boldsymbol{v}^l[0] = \mathbf{0}$ and $\boldsymbol{m}_p^l[0] = \mathbf{0}$.
for $t = 1, \ldots, T$ **do**
  $\boldsymbol{I}^l[t] \leftarrow \boldsymbol{x}^{l-1}[t] + \boldsymbol{m}_p^l[t-1]$.
  $\boldsymbol{m}^l[t] \leftarrow (1 - 1/\tau^l)\boldsymbol{v}^l[t-1] + \boldsymbol{I}^l[t]/\tau^l$.
  $\boldsymbol{s}^l[t] \leftarrow H(\boldsymbol{m}^l[t] - \theta^l)$.
  $\boldsymbol{v}^l[t] \leftarrow \boldsymbol{m}^l[t] - \text{sg}(\boldsymbol{s}^l[t])(\boldsymbol{m}^l[t] - v_{\text{reset}})$.
  $\boldsymbol{\delta}^l[t] \leftarrow a^l \boldsymbol{x}^{l-1}[t] - b^l \boldsymbol{s}^l[t]$.
  $\boldsymbol{m}_p^l[t] \leftarrow (1 - \tau_p^l)\boldsymbol{m}_p^l[t-1] + \tau_p^l \boldsymbol{\delta}^l[t]$.
end for
return $\{\boldsymbol{s}^l[t]\}_{t=1}^{T}$.

Table 9: Feedback-path ablations under the same 256-epoch LIF protocol as Table 5. Metrics are official-test accuracy (%); – denotes an unmeasured configuration.

	SEW ResNet-18		Spiking ResNet-18	
Configuration	Best	Last 5	Best	Last 5
Base LIF	94.90	94.680	93.77	93.450
Input memory only	95.14	94.998	94.96	94.798
Spike feedback only	94.64	94.390	93.96	93.676
Fixed feedback reference	95.36	95.170	94.85	94.572
Detached feedback spike	95.40	95.138	–	–
Learn overall γ	95.13	94.886	95.03	94.870

F Additional Feedback Ablations and Reporting Details

F.1 Protocol and Reference Runs

The new ablations use CIFAR-10 with 50,000 training images and the official 10,000-image test set, LIF neurons, $T = 4$, and batch size 64. Each model is trained for 256 epochs with Adam, initial learning rate 0.001, zero weight decay, and a cosine schedule with horizon 256. MSE compares the time-averaged network output with one-hot labels. All runs start from initialization. The fixed-feedback and overall-gain references reuse earlier 256-epoch runs under the same protocol. An independent check reproduced the initial implementation behavior and the first training epoch of the fixed-feedback reference.

The coefficient-ablation source uses random 32×32 crops with padding 4, horizontal flips, the implementation's CIFAR10Policy, tensor conversion, channel normalization, and one Cutout mask of length 16, in that order. Normalization uses means $(0.4914, 0.4822, 0.4465)$ and standard deviations $(0.2023, 0.1994, 0.2010)$; testing uses only tensor conversion and normalization. Training shuffles data and drops incomplete batches. LIF uses $\tau = 2$, threshold 1, zero initial states and hard reset to zero, with the reset spike detached. The sigmoid surrogate has steepness $k = 4$ in Eq. equation 16. The memory rate starts at $\tau_p = 0.5$ through a zero-initialized sigmoid parameter. Learned coefficients use the same learning rate as network weights, without coefficient clipping; their initialization is given in Appendix.

Best is the maximum of 256 official-test evaluations, and Last 5 is the mean of epochs 252–256 within the same run. Table 5 reports the comparisons; the broader benchmark tables report the reference experiments.

F.2 Input Memory, Spike Feedback and Gradient Ablations

The input-only variant in Table 9 sets $a^l = 1, b^l = 0$; the spike-only variant sets $a^l = 0, b^l = 2$. The full numerical reference sets $a^l = 1, b^l = 2$. The detached-feedback variant retains the full forward recurrence but substitutes $\mathrm{sg}(\boldsymbol{s}^l[t])$ only in the auxiliary update. The overall-gain variant uses $\boldsymbol{I}^l[t] = \boldsymbol{x}^{l-1}[t] + \gamma^l \boldsymbol{m}_p^l[t-1]$ with the original state update, where $\gamma^l = 2\sigma(g^l)$ is initialized at one. All feedback variants retain the learned τ_p^l. The overall-gain variant scales the combined contribution of memory and spike feedback.

The full feedback reference improves Last 5 accuracy over input memory alone by 0.172 percentage points on SEW ResNet-18 but decreases it by 0.226 points on Spiking ResNet-18. Spike-only feedback is likewise architecture dependent. On SEW ResNet-18, retaining the feedback-spike gradient gives 95.170% versus 95.138% with detachment, a difference of 0.032 points. Together, the path ablations show architecture-dependent contributions from input memory, spike feedback and its gradient pathway.

F.3 Broader Exploration and Parameter Diagnostics

Beyond the two LIF comparisons, we completed learn-b^l runs for all 20 combinations of five architectures and four neuron types at $T = 4$. Relative to the historical fixed-feedback logs, 11

Table 10: Broader $T = 4$ exploration: 256-epoch best official-test accuracy (%). The Historical fixed column contains previously recorded fixed-feedback results.

Network	Neuron	Historical fixed	Learn b^l	Δ (pp)
CIFAR10Net	IF	94.24	94.42	+0.18
	LIF	93.84	93.56	-0.28
	PLIF	94.39	94.29	-0.10
	CLIF	94.70	94.61	-0.09
SEW ResNet-18	IF	94.20	94.51	+0.31
	LIF	95.04	95.58	+0.54
	PLIF	95.25	95.08	-0.17
	CLIF	95.42	95.24	-0.18
SEW ResNet-34	IF	95.59	95.60	+0.01
	LIF	95.86	96.11	+0.25
	PLIF	95.87	95.99	+0.12
	CLIF	96.11	96.05	-0.06
Spiking ResNet-18	IF	94.98	95.17	+0.19
	LIF	95.05	95.18	+0.13
	PLIF	95.05	94.99	-0.06
	CLIF	95.29	95.45	+0.16
Spiking ResNet-34	IF	94.67	94.71	+0.04
	LIF	92.43	92.02	-0.41
	PLIF	93.48	93.92	+0.44
	CLIF	95.04	94.95	-0.09

configurations improve and nine decrease in best accuracy, with a descriptive mean difference of 0.0465 percentage points.

The exponential parameterization can also produce large coefficients compensated by small memory update rates. In the completed Spiking ResNet-18 LIF learn-b^l run, one layer has $b^l \approx 86201.5$ and $\tau_p^l \approx 0.000113177$, giving $\tau_p^l b^l \approx 9.756$. In SEW ResNet-18, the largest $b^l \approx 310.906$ is paired with $\tau_p^l \approx 0.0306037$. The paired values characterize both the memory time scale and the effective spike-update magnitude $\tau_p^l b^l$; state trajectories provide a complementary view of their dynamical effect.

G Additional Neuron Model Details

The LIF dynamics and shared notation are defined in Section 3. This appendix records the alternative reset and neuron variants used in the experiments.

Soft reset. Unlike hard reset in Eq. equation 4, soft reset subtracts the threshold after a spike:

$$\boldsymbol{v}^l[t] = \boldsymbol{m}^l[t] - \boldsymbol{\theta}^l \boldsymbol{s}^l[t]. \tag{19}$$

IF, PLIF, and CLIF. The Integrate-and-Fire (IF) neuron is a simplified variant of the LIF model that omits membrane potential leakage. Its update rule replaces Eq. equation 2 with:

$$\boldsymbol{m}^l[t] = \boldsymbol{v}^l[t-1] + \boldsymbol{I}^l[t]. \tag{20}$$

This model assumes perfect integration of inputs until a spike is emitted, making it computationally efficient but less biologically realistic.

The Parametric Leaky Integrate-and-Fire (PLIF) neuron extends the LIF model by treating the membrane time constant τ^l as a learnable parameter rather than a fixed hyperparameter. This allows the network to adapt the temporal dynamics of each neuron during training, enhancing its capacity to capture diverse temporal patterns and improving overall learning performance.

The Complementary Leaky Integrate-and-Fire (CLIF) neuron introduces an auxiliary complementary membrane potential to enrich neuronal dynamics. Specifically, CLIF employs soft reset and removes input decay. Its dynamics are defined as:

$$\boldsymbol{m}^l[t] = (1 - \frac{1}{\tau^l})\boldsymbol{v}^l[t-1] + \boldsymbol{I}^l[t], \tag{21}$$

$$\boldsymbol{s}^l[t] = H(\boldsymbol{m}^l[t] - \theta^l), \tag{22}$$

$$\boldsymbol{m}_c^l[t] = \boldsymbol{m}_c^l[t-1] \cdot \sigma(\frac{1}{\tau^l}\boldsymbol{m}^l[t]) + \boldsymbol{s}^l[t], \tag{23}$$

$$\boldsymbol{v}^l[t] = \boldsymbol{m}^l[t] - \boldsymbol{s}^l[t](\theta^l + \sigma(\boldsymbol{m}_c^l[t])), \tag{24}$$

where $\boldsymbol{m}_c^l[t]$ is a complementary membrane state that accumulates spike history modulated by a sigmoid gate $\sigma(\cdot)$, and the reset magnitude is dynamically adjusted based on both the threshold and this complementary state.