Pseudo-Spiking Neurons: A Noise-Based Training Framework for Heterogeneous-Latency Spiking Neural Networks

Yuxuan Zhang[1], **Yuhang Sun**[1, 2], **Hongjue Li**[1], **Yue Deng**[1, 2], **Wen Yao**[3*]
[1]Beihang University
[2]Beijing Zhongguancun Academy
[3]Defense Innovation Institute, Chinese Academy of Military Science
{zyxuaan, sunyuhang, lihongjue, ydeng}@buaa.edu.cn, wendy0782@126.com

Abstract

Spiking Neural Networks (SNNs) promise significant energy efficiency by processing information via sparse, event-driven spikes. However, realizing this potential is hindered by the conventional use of a rigid, uniform timestep, T. This constraint imposes a challenging trade-off between accuracy and latency, while also incurring the prohibitive training costs of Backpropagation Through Time (BPTT). To overcome this limitation, we introduce the Pseudo-Spiking Neuron (PseudoSN), a novel training proxy that conceptualizes latency as an intrinsic, learnable parameter for each neuron. Building on the efficiency of rate-based methods, the PseudoSN models temporal dynamics in a single, BPTT-free pass. It employs a learnable probabilistic noise scheme to emulate the discretization effects of spike generation (e.g., clipping and quantization), making the neuron-specific timestep—and thus latency—directly optimizable via backpropagation. Integrated into a hardware-aware objective, our framework trains heterogeneous-latency SNNs that autonomously learn to optimize the trade-offs among accuracy, latency and energy, establishing a new state-of-the-art on major benchmarks.

Introduction

Spiking Neural Networks (SNNs), recognized as the third generation of neural network models (Maass 1997), mimic the brain's information processing mechanisms to promise exceptional energy efficiency on neuromorphic hardware (Deng et al. 2020; Davies et al. 2018). By leveraging event-driven, sparse computations, SNNs avoid the costly matrix multiplications of conventional Artificial Neural Networks (ANNs), making them an ideal candidate for power-constrained edge devices and real-time applications.

However, this potential is fundamentally challenged by the temporal dynamics inherent to SNNs. The performance of an SNN is critically dependent on its simulation duration—the number of timesteps (T). While a larger T allows for finer temporal resolution and thus higher accuracy (Kim et al. 2023; Nguyen et al. 2025; Han, Liu, and Karimi 2025), it also incurs greater inference latency and energy consumption, undermining the very advantages of the SNN paradigm. This highlights the number of timesteps, T, as a fundamental and challenging trade-off factor between accuracy and efficiency (Chowdhury, Rathi, and Roy 2021; Dampfhoffer et al. 2023).

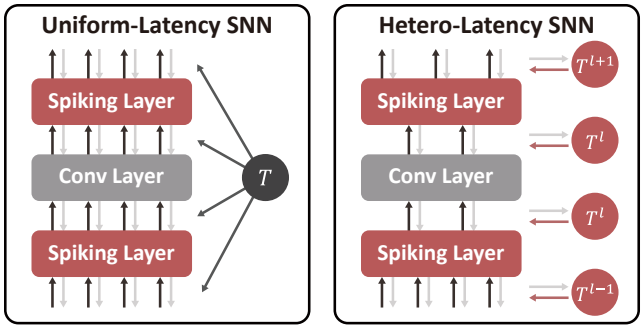

Figure 1: A comparison of latency paradigms in SNNs. **Left**: A uniform-latency SNN uses a fixed global timestep (T) for all layers. **Right**: A hetero-latency SNN employs learnable, layer-specific timesteps (T^l), optimizing each layer's temporal sensitivity for more efficient network configurations.

Despite its critical role, existing SNN training methodologies treat the timestep T as a manually selected, global hyperparameter, imposing a rigid and uniform temporal resolution across the entire network. This "one-size-fits-all" approach creates significant bottlenecks. Conversion-based methods (Hao et al. 2023; Li et al. 2024; Jiang et al. 2024), for instance, often require a large, fixed T to minimize approximation errors from the source ANN, making a search for the optimal T computationally infeasible. Similarly, direct training methods (Wu et al. 2019; Su et al. 2023; Zhang et al. 2023) that rely on Backpropagation Through Time (BPTT) are burdened by a training complexity of $O(LT)$, where a large, pre-set T amplifies the computational cost. Moreover, this uniform latency paradigm is not only inefficient. It is also biologically implausible, ignoring compelling evidence that the brain processes information using heterogeneous temporal dynamics, with different neural pathways operating at varying speeds (Spitmaan et al. 2020; Gao et al. 2020). These insights suggest that an optimal and efficient SNN should not be constrained by a single global timestep but instead should feature a task-specific, heterogeneous latency profile.

To break this rigid paradigm, we argue that a neuron's op-

*Corresponding author.

erational latency should not be a fixed hyperparameter, but an **intrinsic, learnable property** co-optimized with the task objective. We introduce the **Pseudo-Spiking Neuron (PseudoSN)**, a novel proxy that models the temporal dynamics of a spiking neuron within a single, feed-forward pass. The PseudoSN augments a standard activation function with two physically motivated, probabilistic noise terms that emulate the clipping and temporal quantization errors (Bu et al. 2022) inherent in spike-based computation. This BPTT-free approach reduces training complexity from $O(LT)$ to $O(L)$. Critically, the quantization noise is governed by a learnable, neuron-specific timestep parameter, T^l, creating a differentiable pathway to optimize latency. By integrating these parameters into a hardware-aware cost function, our framework trains **Pseudo-Spiking Neural Networks (PseudoSNN)** that learn a heterogeneous latency profile, discovering superior accuracy-latency trade-offs for inference.

The main contributions of this work are as follows:

- The proposal of the Pseudo-Spiking Neuron (PseudoSN), a novel noise-based proxy that enables highly efficient training of deep SNNs.
- A noise-based framework for learning neuron-specific timesteps, leading to the development of heterogeneous-latency PseudoSNN, a new architectural paradigm in neuromorphic computing.
- A differentiable, hardware-aware objective that facilitates the co-optimization of network accuracy, inference latency, and computational energy.
- Comprehensive experimental validation on major benchmarks, demonstrating that our proposed method achieves state-of-the-art performance, yielding superior accuracy at significantly lower latency and computational cost compared to existing SNN optimization techniques.

Related Work

Efficient SNN Training. The standard algorithm for directly training SNNs is Backpropagation Through Time (BPTT) (Lee et al. 2020a), which treats the SNN as a Recurrent Neural Network (RNN) unfolded over time. This approach, while effective, is computationally demanding, with memory and time complexities of $O(LT)$. The high cost of BPTT has spurred the development of more efficient alternatives. **Rate-based backpropagation** methods (Meng et al. 2023; Yu et al. 2024; Yao et al. 2025) assume that for static tasks, information is primarily encoded in the average firing rate, allowing the temporal dynamics to be compressed into a single backward pass with $O(L)$ complexity. **Online training algorithms** (Xiao et al. 2022; Zhu et al. 2024; Hu et al. 2024), inspired by Real-Time Recurrent Learning (RTRL) (Williams and Zipser 1989), also reduce the memory footprint to $O(L)$ by avoiding the need to store the full history of activations, though their time complexity often remains $O(LT)$. Another approach, **Parallel Spiking Neuron** (Fang et al. 2023; Feng et al. 2025), reformulates the neuron dynamics to be non-iterative, enabling parallel computation across the time dimension at the cost of altering the neuron model. Our PseudoSN framework achieves the same favorable $O(L)$ training complexity as rate-based methods but through a different mechanism: we employ a noise-based proxy to approximate the full temporal behavior of a spiking neuron within a single, feed-forward pass.

Latency Optimization for SNNs. A significant limitation of conventionally trained SNNs is their temporal inflexibility; they are optimized for a single, fixed timestep and their performance degrades at other latencies. This has motivated research into several optimization strategies. Methods exist for designing **static heterogeneous latency** (Ding et al. 2024), where different network stages are manually assigned progressively shorter timesteps. Other works focus on **dynamic sample-wise latency** (Li et al. 2023; Wu et al. 2024), where the number of timesteps is adjusted for each input based on its perceived difficulty. A third approach, **mixed-timestep training** (Du et al. 2025; Zhong et al. 2024), trains SNNs on a range of multiple timestep configurations to improve their robustness across various latencies. Our work introduces a different paradigm. Instead of manually designing a latency schedule, enabling dynamic exits, or training for general robustness, our framework learns an optimal, static, and heterogeneous latency configuration while exploring the parameter T at minimal cost and requiring no additional modules during inference.

Noise as a Differentiable Proxy. The core challenge in training SNNs, namely the non-differentiable spike activation, is analogous to training quantized ANNs, which face a non-differentiable rounding function. While the Straight-Through Estimator (STE) (Bengio, Léonard, and Courville 2013) is a common solution, it is often an unstable and biased estimator. An alternative has emerged from quantization literature in the form of **Pseudo-Quantization Noise (PQN)** (Défossez, Adi, and Synnaeve 2022; Shin et al. 2023; Ahn and Yoo 2025), where adding continuous random noise simulates quantization in a fully differentiable manner. We formalize the analogy between spiking and quantization (Bu et al. 2022; Liu et al. 2025): the temporal integration and firing of a neuron over T timesteps is equivalent to value quantization with T levels. We then adapt PQN to model the temporal discretization error inherent in spike generation. This use of noise is fundamentally different from prior noise-based SNN training methods (Wang et al. 2023b; Ma, Yan, and Tang 2023), which inject noise to learn an optimal surrogate gradient function. In contrast, our PseudoSN uses physically motivated noise to directly model the neuron's dynamics (clipping and temporal quantization). Crucially, by parameterizing the noise with a learnable timestep T, we create a dual-purpose mechanism that both facilitates efficient training and enables latency optimization.

Preliminary

Spiking Neuron Models. The Leaky Integrate-and-Fire (LIF) model provides a general framework for neuron dynamics in Spiking Neural Networks (SNNs). For a neuron in layer l at timestep t, the membrane potential u_t^l and spike s_t^l update by:

$$u_t^l = \lambda u_{t-1}^l + W^l s_t^{l-1} - V_{th} s_{t-1}^l, \tag{1}$$

$$s_t^l = H(u_t^l - V_{th}), \tag{2}$$

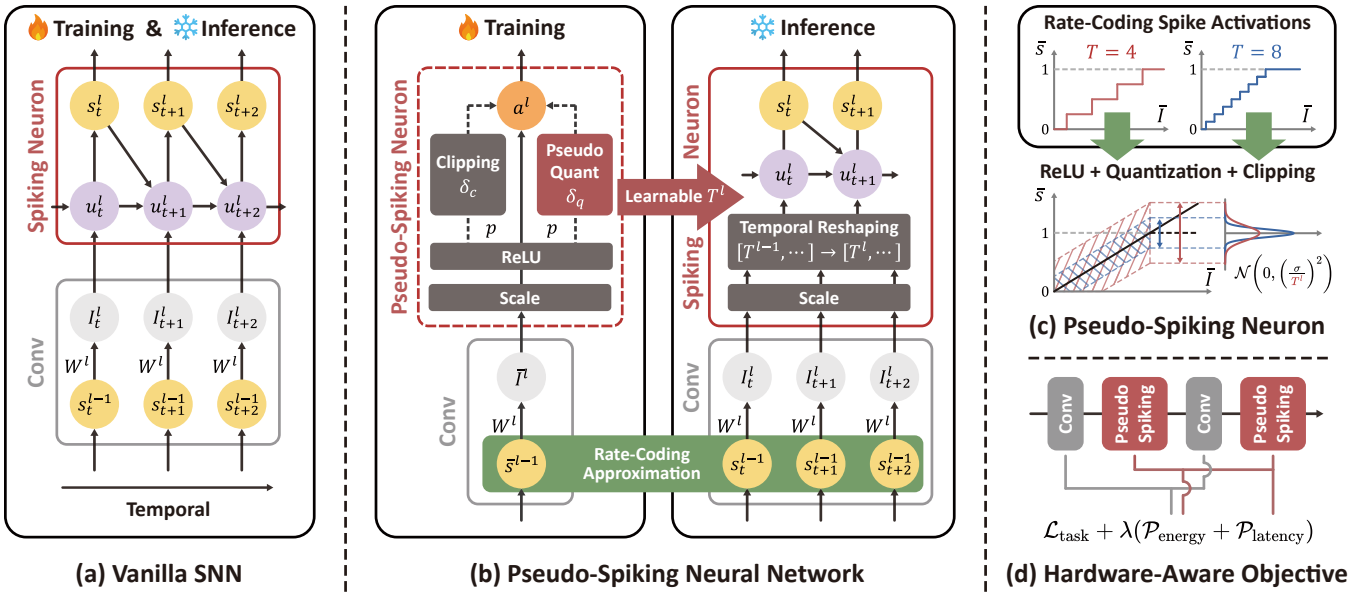

Figure 2: The Proposed **Pseudo-Spiking Neural Network (PseudoSNN)** Framework. **(a)** Vanilla SNN with recurrent dynamics and BPTT. **(b)** PseudoSNN decouples training and inference, using rate-coding and our **Pseudo-Spiking Neuron (PseudoSN)** with learnable timestep T for efficient training, and optimized heterogeneous timesteps for inference. **(c)** The PseudoSN is inspired by the decomposition of rate-coded spike activations into a base ReLU function plus quantization and clipping error terms. **(d)** A hardware-aware Objective penalizes energy and latency to guide the optimization of the learnable timesteps.

Here, the potential integrates weighted inputs ($W^l s_t^{l-1}$), decays by a leak factor λ, and is soft-reset by V_{th} after firing a spike via the Heaviside function $H(\cdot)$. Our work focuses on the widely used and computationally efficient Integrate-and-Fire (IF) model, a special case of the LIF where there is no leak ($\lambda = 1$), making it a perfect integrator of input signals.

Gradient-Based Learning in SNNs. SNNs are typically trained with Backpropagation Through Time (BPTT), which unfolds the recurrent computation. The gradient of the loss $\mathcal{L}$ w.r.t. the membrane potential captures both spatial and temporal dependencies:

$$\frac{\partial \mathcal{L}}{\partial u_t^l} = \frac{\partial \mathcal{L}}{\partial s_t^l}\frac{\partial s_t^l}{\partial u_t^l} + \frac{\partial \mathcal{L}}{\partial u_{t+1}^l}\left(\frac{\partial u_{t+1}^l}{\partial u_t^l} + \frac{\partial u_{t+1}^l}{\partial s_t^l}\frac{\partial s_t^l}{\partial u_t^l}\right). \quad (3)$$

In this equation, the non-differentiable derivative of the spiking function, $\partial s_t^l / \partial u_t^l$, is replaced by a continuous surrogate gradient during the backward pass. The primary limitation of BPTT is the deep temporal dependency chain shown above, which results in prohibitive computational and memory costs.

Method

We introduce PseudoSNN, with a novel spiking neuron that leverages noise-based training to approximate rate-coded spiking dynamics. The framework trains and deploys as a standard SNN with learned heterogeneous latency. Theoretical analysis shows that our noise-driven approach inherently enhances robustness through implicit regularization. Furthermore, we design a latency-aware optimization objective to jointly improve accuracy and hardware efficiency.

From Spiking to Pseudo-Spiking Neurons

The Rate-Based Approximation. The cost of BPTT has motivated the search for more efficient training paradigms. One such approach is rate-coding approximation, which simplifies the gradient calculation by averaging neural activity over the time dimension (Yu et al. 2024).

In standard BPTT, the exact gradient of the loss $\mathcal{L}$ with respect to the weights W^l is computed by summing the contributions from each timestep:

$$\nabla_{W^l}\mathcal{L} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}}{\partial u_t^l}(s_t^{l-1})^\top. \quad (4)$$

Rate-coding approximation provides an approximation to this exact gradient. It first computes the time-averaged presynaptic spike train:

$$\bar{s}^{l-1} = \frac{1}{T}\sum_{t=1}^{T} s_t^{l-1}. \quad (5)$$

The gradient of weights is then calculated based on the loss with respect to the time-averaged input current, $\bar{I}^l = W^l \bar{s}^{l-1}$. This results in the following simplified gradient approximation:

$$(\nabla_{W^l}\mathcal{L})_{\text{rate}} = \frac{\partial \mathcal{L}}{\partial \bar{I}^l}(\bar{s}^{l-1})^\top \approx \nabla_{W^l}\mathcal{L}. \quad (6)$$

By operating on time-averaged values, this method effectively decouples the complex temporal dependencies shown in Eq. (3), leading to a more efficient training process.

Rate-Coding as a Quantization Process. Beyond its computational efficiency, the rate-based approach reveals a fundamental insight: the time-averaged output $\bar{s}^l$ (i.e., the firing rate) is discrete, constrained to the finite set $\{0, 1/T, 2/T, \ldots, 1\}$. For a soft-reset IF neuron, this characteristic effectively converts the input-output relationship into a uniform quantization function:

$$\bar{s}^l = \begin{cases} 0, & \bar{I}^l \leq 0 \\ \lfloor \bar{I}^l \cdot T \rceil / T, & 0 < \bar{I}^l < 1 \\ 1, & \bar{I}^l \geq 1, \end{cases} \tag{7}$$

where $\lfloor \cdot \rceil$ denotes rounding to the nearest integer. This perspective is foundational to major ANN-to-SNN conversion techniques. The discrepancy between an ideal continuous activation (e.g., ReLU) and this quantized output gives rise to two error sources: **quantization error** (the middle case) and **clipping error** (the final case).

Drawing inspiration from classical signal processing, we approximate the quantization error as additive noise. This approach, known as Pseudo-Quantization Noise (PQN), treats the uniform quantizer's error as a zero-mean random variable. Consequently, we can reformulate the quantization function in Eq. 7 as a noise-perturbed identity mapping:

$$\hat{s}^l = \begin{cases} 0, & \bar{I}^l \leq 0 \\ \bar{I}^l + \epsilon \cdot T, & 0 < \bar{I}^l < 1 \\ 1, & \bar{I}^l \geq 1, \end{cases} \tag{8}$$

where ϵ represents the noise approximation. This key formulation replaces the non-differentiable rounding operation with a continuous function augmented by a stochastic component, enabling gradient-based optimization of the quantization step-size parameter T.

The Pseudo-Spiking Neuron Formulation. To consolidate both error sources identified above—quantization error and clipping error—we formulate the Pseudo-Spiking Neuron (PseudoSN) as a unified stochastic model that captures both effects during training (Figure 2c). The PseudoSN operates on the time-averaged input current $\bar{I}^l$ and is constructed from three additive components:

1. **Base Activation and Scaling**: We start with a base continuous activation. To enhance representational capacity and improve gradient flow, we use a ReLU function with learnable affine transformation parameters (γ^l, β^l):

$$\hat{a}^l = \mathrm{ReLU}(\gamma^l \bar{I}^l + \beta^l). \tag{9}$$

   Here, $\hat{a}^l$ represents the idealized, continuous activation value that the discrete firing rate $\bar{s}^l$ approximates.

2. **Modeling Quantization Noise**: We now formalize the quantization error, abstractly represented by ϵ above, as additive PQN. Based on the Central Limit Theorem, the aggregated effect of small, independent temporal discretization errors approaches a Gaussian distribution. We therefore model PQN as zero-mean Gaussian noise, δ_q^l, whose variance is explicitly parameterized by the learnable timestep T^l:

$$\delta_q^l \sim \mathcal{N}\left(0, \left(\frac{\sigma}{T^l}\right)^2\right), \tag{10}$$

   where σ is a fixed hyperparameter. This formulation creates a direct, differentiable link: a higher temporal resolution (larger T^l) inherently reduces the injected noise, forcing the model to allocate latency only where necessary.

3. **Modeling Clipping Error**: The saturation of a neuron's firing rate at its maximum value (1) is modeled explicitly. We define the clipping error as the "clipped-off" portion of the base activation:

$$\delta_c^l = \mathrm{ReLU}(\hat{a}^l - 1). \tag{11}$$

These components are integrated into a single stochastic neuron output. To ensure training stability, the noise and clipping terms are applied via **stochastic gates**:

$$\begin{aligned} a^l &= \hat{a}^l + b_q^l \delta_q^l - b_c^l \delta_c^l, \\ &\text{s.t. } b_q^l, b_c^l \sim \mathrm{Bernoulli}(p), \end{aligned} \tag{12}$$

where the Bernoulli random variables b_q^l and b_c^l apply their respective terms with a probability p.

PseudoSNN Training and Inference Pipeline

Training, Calibration, and Inference. The network is first trained using the PseudoSN as a differentiable proxy. After training, a brief calibration phase is performed. In this phase, each PseudoSN is replaced by a standard soft-reset IF neuron that absorbs the learned scaling parameters (Figure 2b), after which network weights are fine-tuned for few epochs with a small learning rate. This phase is optional to compensate for a statistical mismatch, primarily affecting Batch Normalization (BN) layers. This discrepancy arises because the BN layers learn statistics based on rate-coded activations during training, which differ from the temporal statistics of vanilla spiking neurons encountered during inference (Yu et al. 2024).

Temporal Reshaping for Heterogeneous Latency. During inference, the network operates with a learned heterogeneous latency profile, where layer l processes inputs over T^l timesteps. Temporal reshaping aligns feature maps between layers with different latencies (T^l). Given a preceding layer's output $\mathbf{I}^{l-1}$, the current layer's input $\mathbf{X}^l$ is computed as:

$$\mathbf{X}^l = \begin{cases} \mathbf{I}^{l-1}, & \text{if } T^l = T^{l-1}, \\ \bar{\mathbf{I}}^{l-1}, & \text{otherwise}, \end{cases} \tag{13}$$

where $\bar{\mathbf{I}}^{l-1}$ is the temporal average of $\mathbf{I}^{l-1}$, broadcast across T^l steps. This maintains information flow across layers with varying latencies.

Theoretical Analysis

We provide a theoretical analysis showing that optimizing the stochastic PseudoSN objective is approximately equivalent to minimizing the loss of the deterministic base network, augmented by implicit regularizers that promote robustness to SNN dynamics. Our analysis relies on standard assumptions: the loss function is sufficiently smooth, the stochastic noise and gating variables are independent, and for tractability, we rely on a diagonal approximation of the Hessian.

Proposition 1 *Let $\mathcal{L}_{base}$ be the loss of the deterministic base network. Training with the PseudoSN objective, $\mathcal{L}_{pseudo} = \mathbb{E}[\mathcal{L}(\hat{a}+\delta)]$, is approximately equivalent to minimizing:*

$$\mathcal{L}_{pseudo} \approx \mathcal{L}_{base} \underbrace{-p\sum_{l=1}^{L}(\delta_c^l)^\top \nabla_{\hat{a}^l}\mathcal{L}}_{\textit{Clipping Regularizer}} + \underbrace{\frac{p}{2}\sum_{l=1}^{L}\left(\left(\frac{\sigma}{T^l}\right)^2 + (\delta_c^l)^2\right)^\top \textit{diag}\left(\nabla^2_{\hat{a}^l}\mathcal{L}\right)}_{\textit{Robustness Regularizer}}. \tag{14}$$

Proof Sketch 1 *The regularized objective is derived by substituting the first and second moments of the total perturbation, $\delta^l = b_q^l\delta_q^l - b_c^l\delta_c^l$, into a second-order Taylor expansion of the loss $\mathcal{L}$ around the unperturbed activations $\hat{a}$. See the supplementary materials for the full proof.*

The analysis reveals two implicit regularizers. The first-order **Clipping Regularizer** penalizes activations that exceed the saturation threshold, directly mitigating the clipping error inherent in rate-coded SNNs. The second-order **Robustness Regularizer** penalizes high curvature in the loss landscape, guiding the optimization toward flatter, more robust minima. Crucially, the magnitude of this penalty is scaled by the pseudo-quantization noise variance $(\sigma/T^l)^2$. This creates a differentiable link between a layer's temporal resolution T^l and its required robustness, forcing the network to allocate more latency (larger T^l) only to layers that are more sensitive to quantization.

Hardware-Aware Objective

To co-optimize task accuracy and hardware efficiency, we formulate a total loss function that combines the standard task loss $\mathcal{L}_{\text{task}}$ (e.g., cross-entropy) with a differentiable, hardware-aware regularization term governed by the learnable timesteps T^l (Figure 2d). The final optimization objective is:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{task}} + \lambda(\mathcal{P}_{\text{energy}} + \mathcal{P}_{\text{latency}}), \tag{15}$$

where λ is a hyperparameter balancing the trade-off. The regularization proxies, $\mathcal{P}_{\text{energy}}$ and $\mathcal{P}_{\text{latency}}$, are defined as:

1. **Energy Consumption Proxy**: We estimate energy via Synaptic Operations (SOPs), which scale with a layer's FLOPs (F^l) and spike counts (Horowitz 2014; Rathi and Roy 2021). Since spike counts are positively correlated with the learned timestep T^l, we use it as a differentiable proxy for simplicity, formulating the energy cost as the average product of normalized FLOPs ($n^l = F_l/\max_k F_k$) and T^l:

$$\mathcal{P}_{\text{energy}} = \frac{1}{L}\sum_{l=1}^{L} n_f^l \cdot T^l. \tag{16}$$

2. **Inference Latency Proxy**: The network's overall latency is directly related to its timesteps (Chowdhury, Rathi, and Roy 2022). We define this cost as the average learned timestep across all layers:

$$\mathcal{P}_{\text{latency}} = \frac{1}{L}\sum_{l=1}^{L} T^l. \tag{17}$$

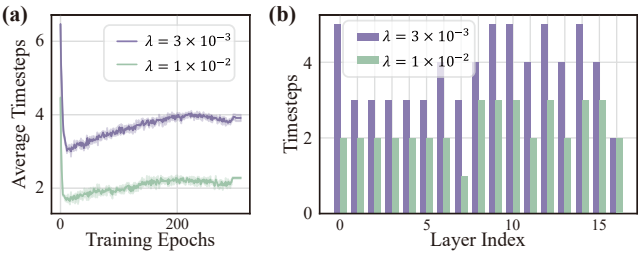

Figure 3: Learned latency for ResNet-18 on CIFAR-100. **(a)** Average timestep convergence during training with different penalty strengths (λ), and **(b)** the final layer-wise timestep distribution.

By combining these proxies, the training objective allows the network to jointly learn task-specific weights and an optimal, heterogeneous latency profile $\{T^l\}$ through end-to-end backpropagation.

Experiments

Implementation Details

To ensure a fair comparison, we evaluate our method on static (CIFAR-10/100 (Krizhevsky and Hinton 2009), ImageNet (Deng et al. 2009)) and neuromorphic (CIFAR10-DVS (Li et al. 2017)) benchmarks using standard VGG and ResNet backbones. For all baseline methods, we adopt the main hyperparameters and data augmentation strategies from RateBP (Yu et al. 2024).

For our PseudoSNN framework, we initialize all learnable timesteps $T^l = 8$ and set the PQN standard deviation to $\sigma = 1/3$ and the noise probability to $p = 0.5$ (linearly annealed from 0.2 to 0.8 on ImageNet for stability). The hardware-aware coefficient λ is set empirically to 3×10^{-3}, 1×10^{-2} and 5×10^{-4} for CIFAR-10, CIFAR-100/ImageNet, and CIFAR10-DVS, respectively. Following training, all models are calibrated for 10 epochs with a learning rate of 1×10^{-3}. Due to the stochastic nature of our method, results are reported as the mean and standard deviation over four independent trials. Further details and additional results are available in the supplementary materials.

State-of-the-Art Performance on Benchmarks

Table 1 shows PseudoSNN achieves the best accuracy-latency balance among competing approaches. The improvement stems from our learned heterogeneous latency allocation, which differs fundamentally from existing paradigms. First, compared to fixed-timestep methods (e.g., BPTT, RateBP), our adaptive approach enables more efficient computation distribution, simultaneously improving accuracy while reducing latency. Second, unlike dynamic

	Method	Architecture	CIFAR10 Top-1 Acc.(T)	CIFAR100 Top-1 Acc.(T)	ImageNet Top-1 Acc.(T)	CIFAR10-DVS Top-1 Acc.(T)
Baseline	BPTT (Lee et al. 2020b)	ResNet-18	95.53(4)	77.72(4)	-	-
		VGG-11	95.61(4)	77.82(4)	-	76.86(10)
		ResNet-19	96.49(4)	81.07(4)	-	-
		SEW-ResNet-34	-	-	67.04(4)	-
Efficient Training	OTTT (Xiao et al. 2022)	VGG-11	93.52±0.06(6)	71.05±0.04(6)	-	76.63±0.34(10)
		PreAct-ResNet-34	-	-	65.15(6)	-
	SLTT (Meng et al. 2023)	ResNet-18	94.44±0.21(6)	74.38±0.30(6)	-	-
		PreAct-ResNet-34	-	-	66.19(6)	-
		VGG-11	-	-	-	77.17±0.23(10)
	SSF (Wang et al. 2023a)	PreAct-ResNet-18	94.90(20)	75.48(20)	-	-
		VGG-11	-	-	-	78.00(20)
	OS (Zhu et al. 2024)	VGG-11	94.35(4)	76.48(4)	-	-
		ResNet-19	95.20(4)	77.86(4)	-	-
		SEW-ResNet-34	-	-	64.14(4)	-
		PreAct-ResNet-34	-	-	67.54(4)	-
	RateBP (Yu et al. 2024)	ResNet-18	95.42±0.11(4)	77.73±0.28(4)	-	-
		VGG-11	95.57±0.08(4)	77.87±0.35(4)	-	76.96±0.13(10)
		ResNet-19	96.26±0.03(4)	80.71±0.12(4)	-	-
		SEW-ResNet-34	-	-	65.84(4)	-
		PreAct-ResNet-34	-	-	69.58(4)	-
Temporal Flexibility	SEENN (Li et al. 2023)	ResNet-19	96.01(1.08)	80.23(1.21)	-	-
	MTT (Du et al. 2025)	ResNet-19	**96.75(4)**	**81.51(4)**	-	-
			96.62(3)	81.14(3)	-	-
		ResNet-34	-	-	67.54(4)	-
			-	-	67.58(3)	-
		VGG-11	-	-	-	75.20(10)
	PseudoSNN (Ours)	VGG-11	**95.80±0.04** (3.22±0.12)	**78.50±0.16** (3.28±0.06)	-	**79.82±0.79** (4.50±0.14)
		ResNet-18	**95.94±0.12** (3.91±0.10)	**78.66±0.25** (4.01±0.06)	-	-
		ResNet-19	96.62±0.02 (3.46±0.05)	81.29±0.11 (3.01±0.03)	-	-
		SEW-ResNet-34	-	-	**68.18±0.25** (4.03±0.29)	-
		PreAct-ResNet-34	-	-	**69.78±0.17** (4.23±0.10)	-

Table 1: Top-1 accuracy (%) comparison with previous works on CIFAR10, CIFAR100, ImageNet and CIFAR10-DVS datasets. T denotes timesteps. Best results are in bold. - indicates results not available (not reported in original article or cannot reproduce). All notations are consistent across experimental conditions for clarity.

methods (e.g., SEENN, MTT) that require expensive training procedures to handle variable latencies, our static optimization achieves comparable flexibility during training at lower computational cost, while eliminating runtime overhead during inference. These advantages collectively produce state-of-the-art performance with practical benefits.

Analysis of Heterogeneous Latency Profiles

We analyzed the learned timesteps of a ResNet-18 on CIFAR-100 to understand its layer-specific timestep allocation (Figure 3). During training, the task loss $\mathcal{L}_{\text{task}}$ in our optimization objective inherently favors higher timesteps, while the regularization term, $\mathcal{P}_{\text{energy}}$ and $\mathcal{P}_{\text{latency}}$, strives to minimize the number of timesteps to reduce the overall computational cost. Thus the average timestep first drops to aggressively prioritize latency reduction, and then rises to preserve accuracy. The final convergence signifies that the network has successfully achieved an optimal accuracy-latency trade-off, aligning with our motivation. This resulting balance point is influenced by the parameter λ, which dictates the network's preference during the learning process. The final layer-wise distribution is U-shaped. Specifically, the network allocates more temporal resolution to early layers for fine-grained feature extraction and to final layers for robust decision integration, while conserving energy in the intermediate layers where representations are more abstract. This

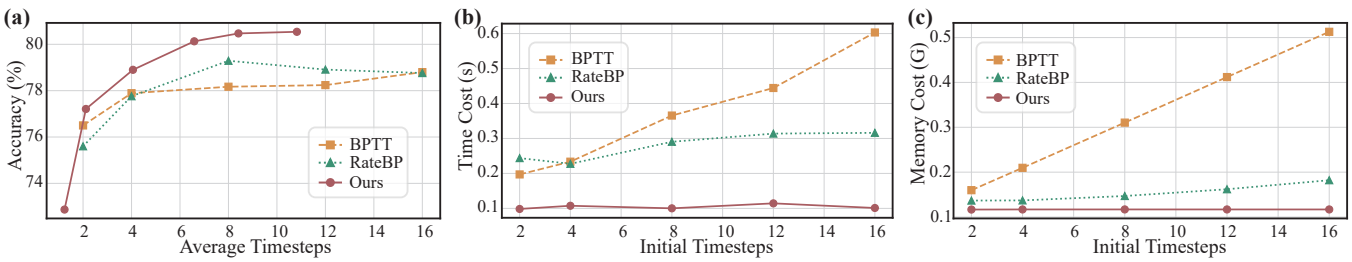

Figure 4: Ablation study for ResNet-18 on CIFAR-100. **(a)** Accuracy-latency Pareto frontier, showing the superiority of our method over BPTT and RateBP. **(b-c)** Training time and peak memory consumption.

	BPTT	RateBP	PseudoSNN			
			base	$+\delta_q$	$+\delta_c$	full
w/o calibration	77.89	77.76	1.60	1.59	31.16	79.02
w/ calibration	77.93	77.95	34.71	35.81	54.79	79.57

Table 2: Ablation study of PseudoSNN's noise components on CIFAR-100. The "w/ calibration" condition includes a 10-epoch fine-tuning phase.

non-uniform allocation indicates the network learns an efficient, hierarchical processing strategy.

Ablation Studies

Accuracy-Latency Pareto Frontier. We first evaluated the accuracy-latency trade-off using ResNet-18 on CIFAR-100. By adjusting λ, our method traces a Pareto frontier that consistently dominates both BPTT and RateBP, achieving higher accuracy for any given latency, as shown in Figure 4a.

Training Efficiency. As shown in Figures 4b and 4c, our method demonstrates superior training efficiency. While both our approach and RateBP decouple training cost from the number of timesteps by leveraging a rate-coding approximation (which enables a single-pass gradient), our computationally simpler PseudoSNN proxy results in significantly lower training time and peak memory consumption compared to both baselines.

Contribution of Noise Components The ablation study in Table 2 demonstrates the complementary roles of the PQN (δ_q) and clipping (δ_c) noise components. While models incorporating only a single noise term fail in training, their combination leads to high accuracy, underscoring that both are essential. Furthermore, the calibration phase is optional, as its impact is limited (less than 1% improvement).

Energy Efficiency Table 3 presents the energy efficiency analysis for the SEW-ResNet-34 architecture on ImageNet. Following established methodologies (Rathi and Roy 2021; Sengupta et al. 2019), we estimate energy consumption based on the number of Synaptic Operations (SOPs). Compared to the SEW-ResNet-34 baseline (Fang et al. 2021)

Method	Time Step	SOPS (G) (↓)	Energy (mJ) (↓)	Top1 Acc. (↑)
ResNet-34	1	3.66	16.85	73.29
SEW-ResNet-34	4	4.95 (35.01%↑)	4.89 (70.99%↓)	66.58 (-6.71)
PseudoSNN (Ours)	4.09	**4.10 (11.9%↑)**	**4.13 (75.5%↓)**	**68.18 (-5.11)**

Table 3: Energy efficiency comparison on ImageNet. Our method is benchmarked against an ANN (ResNet-34) and an SNN baseline (SEW-ResNet-34). Values in brackets denote the change relative to the ANN baseline.

which uses a uniform timestep, our method with heterolatency achieves higher accuracy with noticeably lower SOPs and energy consumption at a similar average timestep ($\sim$4). This superior efficiency stems from a learned hardware-aware allocation strategy that optimizes the comprehensive Pareto frontier of accuracy, latency, and energy.

Conclusion

In this paper, we addressed the critical limitations of Spiking Neural Networks (SNNs)—namely, the high computational cost of BPTT-based training and the inflexibility of uniform latency. We introduced the Pseudo-Spiking Neuron (PseudoSN), a novel, BPTT-free training proxy that reframes latency as an intrinsic, learnable property of individual neurons. The PseudoSN models the temporal dynamics of a spiking neuron in a single feed-forward pass by injecting physically motivated, probabilistic noise to simulate clipping and temporal quantization errors. Crucially, by parameterizing the quantization noise with a learnable, neuron-specific timestep, our framework enables end-to-end optimization of a heterogeneous latency profile, guided by a hardware-aware objective. Comprehensive experiments demonstrate that our method noticeably reduces training complexity and establishes a new state-of-the-art Pareto frontier for accuracy and latency. Our work paves the way for the development of more scalable, efficient, and deployment-friendly SNNs for neuromorphic computing, with future extensions to more complex architectures and validated on neuromorphic processors.

Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grant 62325101, Grant 62031001 and Grant 62405014.

References

Ahn, M.; and Yoo, S. 2025. Gaussian Weight Sampling for Scalable, Efficient and Stable Pseudo-Quantization Training. *arXiv preprint arXiv:2505.11170*.

Bengio, Y.; Léonard, N.; and Courville, A. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

Bu, T.; Fang, W.; Ding, J.; DAI, P.; Yu, Z.; and Huang, T. 2022. Optimal ANN-SNN Conversion for High-accuracy and Ultra-low-latency Spiking Neural Networks. In *International Conference on Learning Representations*.

Chowdhury, S. S.; Rathi, N.; and Roy, K. 2021. One timestep is all you need: Training spiking neural networks with ultra low latency. *arXiv preprint arXiv:2110.05929*.

Chowdhury, S. S.; Rathi, N.; and Roy, K. 2022. Towards ultra low latency spiking neural networks for vision and sequential tasks using temporal pruning. In *European Conference on Computer Vision*, 709–726. Springer.

Dampfhoffer, M.; Mesquida, T.; Valentian, A.; and Anghel, L. 2023. Backpropagation-based learning techniques for deep spiking neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 35(9): 11906–11921.

Davies, M.; Srinivasa, N.; Lin, T.-H.; Chinya, G.; Cao, Y.; Choday, S. H.; Dimou, G.; Joshi, P.; Imam, N.; Jain, S.; et al. 2018. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro*, 38(1): 82–99.

Défossez, A.; Adi, Y.; and Synnaeve, G. 2022. Differentiable Model Compression via Pseudo Quantization Noise. *Transactions on Machine Learning Research*.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.

Deng, L.; Wang, G.; Li, G.; Li, S.; Liang, L.; Zhu, M.; Wu, Y.; Yang, Z.; Zou, Z.; Pei, J.; et al. 2020. Tianjic: A unified and scalable chip bridging spike-based and continuous neural computation. *IEEE Journal of Solid-State Circuits*, 55(8): 2228–2246.

Ding, Y.; Zuo, L.; Jing, M.; He, P.; and Xiao, Y. 2024. Shrinking your timestep: Towards low-latency neuromorphic object recognition with spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 11811–11819.

Du, K.; Wu, Y.; Deng, S.; and Gu, S. 2025. Temporal Flexibility in Spiking Neural Networks: Towards Generalization Across Time Steps and Deployment Friendliness. In *The Thirteenth International Conference on Learning Representations*. Singapore.

Fang, W.; Yu, Z.; Chen, Y.; Huang, T.; Masquelier, T.; and Tian, Y. 2021. Deep residual learning in spiking neural networks. *Advances in Neural Information Processing Systems*, 34: 21056–21069.

Fang, W.; Yu, Z.; Zhou, Z.; Chen, D.; Chen, Y.; Ma, Z.; Masquelier, T.; and Tian, Y. 2023. Parallel spiking neurons with high efficiency and ability to learn long-term dependencies. *Advances in Neural Information Processing Systems*, 36: 53674–53687.

Feng, W.; Gao, X.; Du, W.; Shi, H.; Zhao, P.; Wu, P.; and Miao, C. 2025. Efficient Parallel Training Methods for Spiking Neural Networks with Constant Time Complexity. In *Forty-second International Conference on Machine Learning*.

Gao, R.; Van den Brink, R. L.; Pfeffer, T.; and Voytek, B. 2020. Neuronal timescales are functionally dynamic and shaped by cortical microarchitecture. *elife*, 9: e61277.

Han, C.; Liu, L.-J.; and Karimi, H. R. 2025. Exploring temporal information dynamics in Spiking Neural Networks: Fast Temporal Efficient Training. *Journal of Neuroscience Methods*, 417: 110401.

Hao, Z.; Bu, T.; Ding, J.; Huang, T.; and Yu, Z. 2023. Reducing ann-snn conversion error through residual membrane potential. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, 11–21.

Horowitz, M. 2014. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE international solid-state circuits conference digest of technical papers (ISSCC)*, 10–14. IEEE.

Hu, J.; Yao, M.; Qiu, X.; Chou, Y.; Cai, Y.; Qiao, N.; Tian, Y.; Xu, B.; and Li, G. 2024. High-Performance Temporal Reversible Spiking Neural Networks with $O(L)$ Training Memory and $O(1)$ Inference Cost. In *Forty-first International Conference on Machine Learning*.

Jiang, Y.; Hu, K.; Zhang, T.; Gao, H.; Liu, Y.; Fang, Y.; and Chen, F. 2024. Spatio-temporal approximation: A training-free snn conversion for transformers. In *The twelfth international conference on learning representations*.

Kim, Y.; Li, Y.; Park, H.; Venkatesha, Y.; Hambitzer, A.; and Panda, P. 2023. Exploring temporal information dynamics in spiking neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, 8308–8316.

Krizhevsky, A.; and Hinton, G. 2009. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*.

Lee, C.; Sarwar, S. S.; Panda, P.; Srinivasan, G.; and Roy, K. 2020a. Enabling spike-based backpropagation for training deep neural network architectures. *Frontiers in neuroscience*, 14: 497482.

Lee, C.; Sarwar, S. S.; Panda, P.; Srinivasan, G.; and Roy, K. 2020b. Enabling spike-based backpropagation for training deep neural network architectures. *Frontiers in neuroscience*, 14: 497482.

Li, H.; Liu, H.; Ji, X.; Li, G.; and Shi, L. 2017. Cifar10-dvs: an event-stream dataset for object classification. *Frontiers in neuroscience*, 11: 244131.

Li, Y.; Deng, S.; Dong, X.; and Gu, S. 2024. Error-aware conversion from ANN to SNN via post-training parameter calibration. *International Journal of Computer Vision*, 132(9): 3586–3609.

Li, Y.; Geller, T.; Kim, Y.; and Panda, P. 2023. Seenn: Towards temporal spiking early exit neural networks. *Advances in Neural Information Processing Systems*, 36: 63327–63342.

Liu, C.; Shen, J.; Ran, X.; Xu, M.; Xu, Q.; Xu, Y.; and Pan, G. 2025. Efficient ANN-SNN Conversion with Error Compensation Learning. In *Forty-second International Conference on Machine Learning*. PMLR.

Ma, G.; Yan, R.; and Tang, H. 2023. Exploiting noise as a resource for computation and learning in spiking neural networks. *Patterns*, 4(10).

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

Meng, Q.; Xiao, M.; Yan, S.; Wang, Y.; Lin, Z.; and Luo, Z.-Q. 2023. Towards memory-and time-efficient backpropagation for training spiking neural networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, 6166–6176.

Nguyen, D. A.; Araya, E.; Fono, A.; and Kutyniok, G. 2025. Time to Spike? Understanding the Representational Power of Spiking Neural Networks in Discrete Time. In *Forty-second International Conference on Machine Learning*.

Rathi, N.; and Roy, K. 2021. Diet-snn: A low-latency spiking neural network with direct input encoding and leakage and threshold optimization. *IEEE Transactions on Neural Networks and Learning Systems*, 34(6): 3174–3182.

Sengupta, A.; Ye, Y.; Wang, R.; Liu, C.; and Roy, K. 2019. Going deeper in spiking neural networks: VGG and residual architectures. *Frontiers in neuroscience*, 13: 95.

Shin, J.; So, J.; Park, S.; Kang, S.; Yoo, S.; and Park, E. 2023. Nipq: Noise proxy-based integrated pseudo-quantization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 3852–3861.

Spitmaan, M.; Seo, H.; Lee, D.; and Soltani, A. 2020. Multiple timescales of neural dynamics and integration of task-relevant signals across cortex. *Proceedings of the National Academy of Sciences*, 117(36): 22522–22531.

Su, Q.; Chou, Y.; Hu, Y.; Li, J.; Mei, S.; Zhang, Z.; and Li, G. 2023. Deep directly-trained spiking neural networks for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 6555–6565.

Wang, J.; Song, Z.; Wang, Y.; Xiao, J.; Yang, Y.; Mei, S.; and Zhang, Z. 2023a. Ssf: Accelerating training of spiking neural networks with stabilized spiking flow. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 5982–5991.

Wang, Z.; Jiang, R.; Lian, S.; Yan, R.; and Tang, H. 2023b. Adaptive smoothing gradient learning for spiking neural networks. In *International conference on machine learning*, 35798–35816. PMLR.

Williams, R. J.; and Zipser, D. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2): 270–280.

Wu, D.; Qi, Y.; Cai, K.; Jin, G.; Yi, X.; and Huang, X. 2024. Direct training needs regularisation: Anytime optimal inference spiking neural network. *arXiv preprint arXiv:2405.00699*.

Wu, Y.; Deng, L.; Li, G.; Zhu, J.; Xie, Y.; and Shi, L. 2019. Direct training for spiking neural networks: Faster, larger, better. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 1311–1318.

Xiao, M.; Meng, Q.; Zhang, Z.; He, D.; and Lin, Z. 2022. Online training through time for spiking neural networks. *Advances in neural information processing systems*, 35: 20717–20730.

Yao, M.; Qiu, X.; Hu, T.; Hu, J.; Chou, Y.; Tian, K.; Liao, J.; Leng, L.; Xu, B.; and Li, G. 2025. Scaling spike-driven transformer with efficient spike firing approximation training. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Yu, C.; Liu, L.; Wang, G.; Li, E.; and Wang, A. 2024. Advancing training efficiency of deep spiking neural networks through rate-based backpropagation. *Advances in Neural Information Processing Systems*, 37: 115786–115815.

Zhang, H.; Li, Y.; He, B.; Fan, X.; Wang, Y.; and Zhang, Y. 2023. Direct training high-performance spiking neural networks for object recognition and detection. *Frontiers in Neuroscience*, 17: 1229951.

Zhong, X.; Hu, S.; Liu, W.; Huang, W.; Ding, J.; Yu, Z.; and Huang, T. 2024. Towards low-latency event-based visual recognition with hybrid step-wise distillation spiking neural networks. In *Proceedings of the 32nd ACM international conference on multimedia*, 9828–9836.

Zhu, Y.; Ding, J.; Huang, T.; Xie, X.; and Yu, Z. 2024. Online stabilization of spiking neural networks. In *The Twelfth International Conference on Learning Representations*.