The Dormant Spiking Neuron: A State-Driven Mechanism for Efficient Spiking Neural Networks

Ruichen Ma[1], **Yicai Chen**[1], **Pujun Zhou**[1], **Yue Zuo**[1], **Chongyang Li**[1], **Shuang Liu**[1], **Shaogang Hu**[1,2], **Guanchao Qiao**[1*]
[1]University of Electronic Science and Technology of China (UESTC)
[2]Shenzhen Institute for Advanced Study, UESTC
ruichen.ma@std.uestc.edu.cn, gcqiao@uestc.edu.cn
https://github.com/Ruichen0424/DSN

Abstract

Spiking neural networks (SNNs) offer a compelling pathway toward energy-efficient artificial intelligence, fundamentally anchored in presynaptic input-driven sparsity. However, a substantial yet chronically overlooked source of computational overhead persists: the continuous subthreshold dynamics of postsynaptic neurons. Quantitative analysis reveals that up to 44% of neurons in an ImageNet-trained ResNet-18 model reside in a deeply inhibited state, imposing a persistent computational burden despite having a negligible probability of firing. To address this inefficiency, the Dormant Spiking Neuron (DSN) model is introduced. By incorporating a state-driven dynamic gate, the DSN allows a neuron to enter a dormant state when its membrane potential falls below a predefined negative threshold, dynamically pruning redundant computations at the instance level. Crucially, this dormancy mechanism is directly integrated into the neuronal dynamics and trained end-to-end via backpropagation through time using surrogate gradients, enabling the network to actively learn a robust dormancy policy that adaptively preserves critical information flow. Extensive experiments across static and neuromorphic datasets demonstrate that the DSN framework consistently reduces computational energy by 40% to over 70%. Notably, on the large-scale ImageNet dataset, the DSN outperforms state-of-the-art static pruning techniques at comparable reported cost reductions and simultaneously improves the corresponding baseline SNN accuracy by 1.34%. By establishing a trainable, state-driven sparsity paradigm orthogonal to conventional input-driven methods, the DSN unlocks synergistic efficiency gains, presenting an effective approach for advancing neuromorphic computing.

1 Introduction

The escalating demand for artificial intelligence on resource-constrained platforms [1, 2, 3, 4, 5] has catalyzed significant research into energy-efficient computing paradigms. Among these, spiking neural networks (SNNs) have emerged as a compelling alternative to conventional artificial neural networks (ANNs), owing to their biologically plausible, event-driven computational nature [6, 7, 8, 9, 10, 11, 12, 13, 14]. Fundamentally, the efficiency of SNNs is anchored in input-driven sparsity, a presynaptic mechanism wherein computations are triggered exclusively by the arrival of discrete spike events. This mechanism effectively bypasses operations associated with inactive inputs, leading to substantial reductions in the total number of synaptic operations compared to the dense matrix multiplications characteristic of ANNs [15, 16, 17, 18, 19].

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Despite this advantage, a substantial yet chronically overlooked source of computational overhead persists within the SNN framework, stemming from the continuous subthreshold dynamics of postsynaptic neurons. Even when a receiving neuron remains quiescent, it still performs membrane-potential integration and leakage calculations at every timestep [20]. To quantify the magnitude of this inefficiency, we analyzed the membrane-potential distribution in a ResNet-18 SNN model trained on the ImageNet dataset. The cumulative distribution function of neuronal membrane potentials reveals three distinct regions (see the Appendix C.1 and fig. 4a for a detailed visualization). While approximately 26% of neurons are in the active spiking region and 30% reside in the subthreshold active region ($0 < V_m < V_{th}$), which is critical for potential future spikes, 44% of the neurons are found in the deeply inhibited region ($V_m \leq 0$). These deeply inhibited neurons have a negligible probability of firing in the immediate future. Consequently, the computational resources consistently dedicated to updating their states represent an opportunity for optimization.

Targeting this specific source of inefficiency, this paper introduces the Dormant Spiking Neuron (DSN) model. By incorporating a state-driven dynamic gate, the DSN allows a neuron to enter a dormant state when its membrane potential falls below a predefined negative threshold, dynamically excluding it from all subsequent computations within that sample. This postsynaptic sparsity operates orthogonally to conventional presynaptic input-driven sparsity, enabling a multiplicative reduction in overall computational load. The DSN differs from conventional static pruning techniques. While traditional network pruning aims to remove structures based on dataset-level, chronic redundancy [21, 22], the DSN leverages state-driven sparsity to target instance-level, transient redundancy. Recent advances in neuromorphic hardware have demonstrated the potential for physical energy savings from threshold-based neuron inhibition mechanisms [23].

However, static, post-hoc thresholding inherently assumes that early inhibition implies future unimportance. Although this approach can be effective for stationary inputs, it may be less robust and can lead to accuracy degradation in complex or nonstationary temporal streams. To address this limitation, the dormancy mechanism of the DSN is directly integrated into the neuronal dynamics and trained end-to-end via backpropagation through time (BPTT) using surrogate gradients (SG). This allows the network to adapt its weight distribution to the dormancy constraint and learn a robust dormancy policy that preserves critical information flow. The primary contributions of this work are summarized as follows:

- The DSN is proposed as a state-driven sparsity paradigm that specifically targets the computational inefficiency of deeply inhibited neurons. By introducing dynamic, instance-level neuronal dormancy, the DSN operates orthogonally to conventional presynaptic input-driven sparsity, thereby enabling a multiplicative reduction in overall synaptic operations.
- The BPTT training methodology for the DSN is formulated using surrogate gradients, thereby mitigating the performance degradation that can arise from static or post-hoc thresholding and allowing the network to learn a robust dormancy policy.
- Extensive experiments on multiple static and neuromorphic datasets demonstrate that the DSN reduces computational energy by 40% to over 70% while maintaining competitive performance. Notably, on the ImageNet dataset, the proposed method outperforms state-of-the-art static pruning techniques at comparable reported cost reductions while simultaneously improving the corresponding baseline SNN accuracy by 1.34%.

2 Related Work

Various pruning techniques aim to improve SNN efficiency by identifying and removing less critical network components based on static properties or observed activity [24, 25, 26, 21, 22, 27]. Common approaches include unstructured pruning, such as eliminating synapses with small weight magnitudes, and structured pruning, which removes entire neurons based on criteria such as output contribution. Within the SNN domain, these methods often use neuronal firing rates, permanently eliminating components that fall below a specified firing-rate threshold. While effective at reducing overall network size, these conventional pruning strategies predominantly target dataset-level, chronic redundancy. By relying on statistical measures of importance derived from representative datasets, these methods have limited adaptability to dynamic inputs, where the relevance of a specific neuron may vary rapidly across samples. Furthermore, decisions based solely on long-term firing rates do

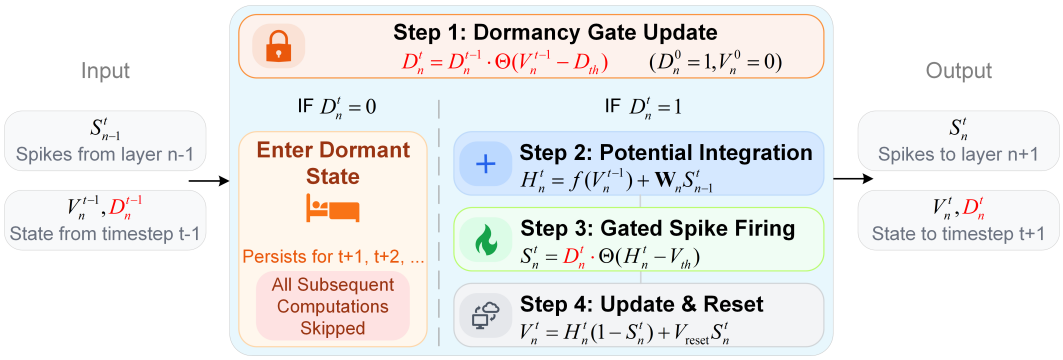


Figure 1: The four-step dynamics of the Dormant Spiking Neuron (DSN). At each timestep t, a DSN executes a sequence of operations. (1) Dormancy Gate Update: The neuron first checks whether its previous membrane potential V_n^{t-1} has fallen below the dormancy threshold D_{th}. If so, it enters a permanent dormant state, and all subsequent stages are bypassed for the current and all future timesteps. (2) Potential Integration: If active ($D_n^t = 1$), the neuron integrates presynaptic inputs with its decayed membrane potential. (3) Gated Spike Firing: A spike S_n^t is generated only if the neuron is active and its integrated potential H_n^t surpasses the firing threshold V_{th}. (4) Potential Update & Reset: The membrane potential V_n^t is updated for the next timestep, incorporating a reset mechanism.

not capture the real-time computational overhead associated with the subthreshold state updates of temporarily inhibited neurons, leaving a substantial portion of computational redundancy unaddressed.

The proposed DSN differs from conventional static pruning by introducing dynamic, state-driven sparsity. Instead of permanently removing structural components, the DSN dynamically bypasses redundant computations when a neuron's membrane potential falls below a predefined negative threshold ($V_m < D_{th}$). This mechanism specifically targets instance-level, transient redundancy, offering flexibility and adaptability compared with statistical pruning methods. Because the DSN regulates postsynaptic computations based on the internal state of the receiving neuron, it operates orthogonally to conventional presynaptic input-driven sparsity. This orthogonality enables the two forms of sparsity to act synergistically, yielding multiplicative reductions in computational cost.

3 Method

3.1 Dormant Spiking Neuron Dynamics

The core of the DSN involves introducing a binary activity state, termed the dormancy gate $D_n^t \in \{0, 1\}$, for each neuron in layer n at timestep t. This gate dynamically determines whether a neuron should enter a permanent state of inactivity, thereby bypassing its associated computations. The complete forward dynamics of a DSN follow a four-stage process at each timestep, as shown in fig. 1.

Step 1: Dormancy Gate Update. The activity state D_n^t is updated based on the membrane potential from the previous timestep V_n^{t-1}. If V_n^{t-1} drops below a predefined dormancy threshold D_{th}, the gate irreversibly closes ($D_n^t \to 0$). This update is described by:

$$D_n^t = D_n^{t-1} \cdot \Theta(V_n^{t-1} - D_{th}) \tag{1}$$

Initially, all neurons start in an active state ($D_n^0 = 1$) with their membrane potential at rest ($V_n^0 = 0$). The multiplicative nature of eq. (1) ensures that once the gate closes ($D_n^t = 0$), it remains permanently closed for all subsequent timesteps ($D_n^{t_0} \equiv 0$ for any $t_0 > t$), rendering the neuron dormant.

Step 2: Potential Integration. Active neurons ($D_n^t = 1$) integrate weighted spikes S_{n-1}^t from the presynaptic layer with their decayed membrane potential $f(V_n^{t-1})$ from the previous timestep:

$$H_n^t = f(V_n^{t-1}) + \mathbf{W}_n S_{n-1}^t \tag{2}$$

Step 3: Gated Spike Firing. A spike S_n^t is generated only if the neuron is active ($D_n^t = 1$) and its integrated potential H_n^t surpasses the firing threshold V_{th}:

$$S_n^t = D_n^t \cdot \Theta(H_n^t - V_{th}) \tag{3}$$

Step 4: Potential Update & Reset. The membrane potential V_n^t for the next timestep is calculated, incorporating a reset mechanism for neurons that have fired:

$$V_n^t = H_n^t(1 - S_n^t) + V_{\text{reset}} S_n^t \tag{4}$$

Here, $\Theta(\cdot)$ is the Heaviside step function, $\mathbf{W}_n$ is the weight matrix from layer $n-1$ to n, and $f(\cdot)$ denotes the membrane potential decay function. For a standard Leaky Integrate-and-Fire (LIF) neuron model, $f(V)$ is typically defined as $f(V) = \tau^{-1} \cdot V$, where τ is the membrane time constant.

Compared to a standard spiking neuron, the DSN incorporates two primary modifications. These are the dormancy gate update mechanism specified in eq. (1) and the gating of the spike output by D_n^t in eq. (3). A crucial consequence of this design is that a dormant neuron will never fire again. Therefore, subsequent dynamics for that neuron, including membrane-potential updates, need not be executed. This property underlies the computational savings provided by the DSN.

3.2 Direct Training via Surrogate Gradients

A primary challenge in training SNNs arises from the non-differentiable nature of the Heaviside step function $\Theta(\cdot)$, which appears in both eqs. (1) and (3). The surrogate-gradient method enables gradient-based training despite the non-differentiability of the Heaviside step function $\Theta(\cdot)$ [28]. The core principle involves using the exact function $\Theta(\cdot)$ in the forward pass but substituting its derivative with a continuous approximation, the surrogate function $\sigma'(\cdot)$, during the backward pass.

This enables the derivation of a complete set of update rules for backpropagation through time. The primary goal of BPTT is to compute the gradient of the loss function L with respect to the weights $\mathbf{W}_n$, which is achieved by summing the contributions from each timestep:

$$\frac{\partial L}{\partial \mathbf{W}_n} = \sum_{t=1}^{T} \frac{\partial L}{\partial H_n^t} \frac{\partial H_n^t}{\partial \mathbf{W}_n} = \sum_{t=1}^{T} \delta_{H,n}^t (S_{n-1}^t)^\top \tag{5}$$

The weight gradient depends on the error term $\delta_{H,n}^t \equiv \frac{\partial L}{\partial H_n^t}$, which represents the gradient of the loss concerning the integrated potential H_n^t. The term $\delta_{H,n}^t$ is part of a system of interconnected gradients that must be determined by propagating errors backward from the network output. To formalize this process, error terms are defined for the other key variables, including $\delta_{S,n}^t$ for the spike output S_n^t, $\delta_{V,n}^t$ for the updated potential V_n^t, and $\delta_{D,n}^t$ for the dormancy gate state D_n^t. These errors propagate backward through spatial layers and temporal steps according to the following recurrence relations:

$$\begin{cases} \delta_{H,n}^t = \delta_{S,n}^t \cdot [D_n^t \cdot \sigma_s'(H_n^t - V_{th})] + \delta_{V,n}^t \cdot (1 - S_n^t) \\ \delta_{S,n}^t = \mathbf{W}_{n+1}^\top \delta_{H,n+1}^t \\ \delta_{V,n}^t = \delta_{D,n}^{t+1} \cdot [D_n^t \cdot \sigma_d'(V_n^t - D_{th})] + \delta_{H,n}^{t+1} \cdot f'(V_n^t) \\ \delta_{D,n}^t = \delta_{D,n}^{t+1} \cdot \Theta(V_n^t - D_{th}) + \delta_{S,n}^t \cdot \Theta(H_n^t - V_{th}) \end{cases} \tag{6}$$

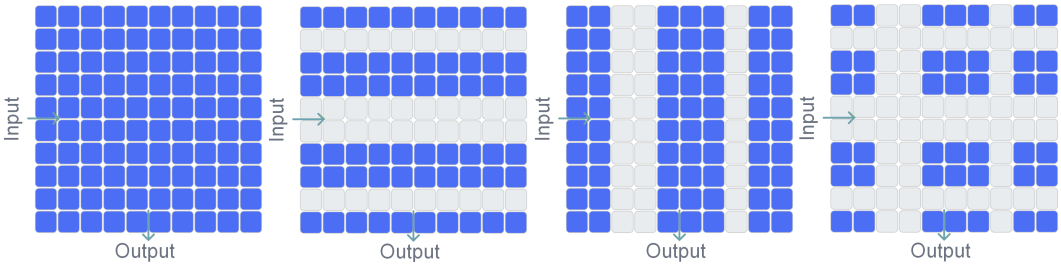
Here, $\sigma_s'(\cdot)$ and $\sigma_d'(\cdot)$ represent the surrogate gradient functions used for spike firing and dormancy gate mechanisms, respectively. The backpropagation process begins at the output layer at the final timestep and iteratively computes gradients that are propagated backward through time and space. Solving this system yields the required $\delta_{H,n}^t$ terms at each timestep, which are then used to compute the final weight gradient via eq. (5). Full mathematical derivations are available in the Appendix A.

3.3 Analysis of Computational Efficiency

The primary advantage of the DSN is its ability to reduce computational cost by exploiting state-driven sparsity that is orthogonal to conventional input-driven sparsity. Computational load is quantified using synaptic operations (SOPs).

Input-Driven Sparsity. Standard SNNs gain efficiency through this presynaptic mechanism. If the input spike rate at timestep t is $\overline{R_t}$, only computations associated with the corresponding rows of the weight matrix $\mathbf{W}_n$ are performed. The total SOPs for a network layer benefiting solely from this principle can be estimated as: $\text{SOPs}_{\text{input-driven}} = \text{MACs} \cdot \sum_{t=1}^{T} \overline{R_t}$.

State-Driven Sparsity. The DSN model introduces an additional mechanism through its dormancy feature. If the activity rate (fraction of non-dormant neurons) of the postsynaptic neurons at timestep



(a) Dense operation. (b) Input-driven sparsity. (c) State-driven sparsity. (d) Orthogonal sparsity.

Figure 2: Orthogonal sparsity mechanisms in the DSN model. (a) In dense ANNs, all operations are active. (b) Traditional SNNs exploit input-driven sparsity, wherein computations involving inactive input rows are skipped. (c) The dormancy mechanism introduces state-driven sparsity, wherein computations involving inactive output columns are skipped. (d) The DSN combines both mechanisms, yielding a multiplicative reduction in the number of active synaptic operations.

t is $\overline{D_t}$, only computations associated with the corresponding columns of the weight matrix $\mathbf{W}_n$ need to be performed. The SOP count for a hypothetical model utilizing only this state-driven approach would be: $\text{SOPs}_{\text{state-driven}} = \text{MACs} \cdot \sum_{t=1}^{T} \overline{D_t}$.

Orthogonal Sparsity in the DSN. Because the two mechanisms are orthogonal, the DSN model combines both simultaneously, allowing their effects to compound multiplicatively as shown in fig. 2. This yields a computational cost lower than that of either sparsity type alone: $\text{SOPs}_{\text{DSN}} = \text{MACs} \cdot \sum_{t=1}^{T} (\overline{R_t} \cdot \overline{D_t})$. By incorporating the dormancy factor $\overline{D_t}$, the DSN can further reduce computational cost beyond the savings offered by input-driven sparsity alone. Detailed network SOPs and energy estimates are provided in the Appendix B.

3.4 Inference-Time Optimization vs. Direct Training

A key feature of the DSN framework is its flexibility, offering two distinct paradigms for application: inference-time optimization (ITO) and direct training via BPTT. The ITO paradigm acts as a plug-and-play optimization tool applied directly after training. However, applying a fixed dormancy rule to a pre-trained network has a key limitation: early inhibition does not necessarily imply future unimportance. For quasi-stationary inputs, ITO effectively bypasses redundant computations with limited accuracy loss. Yet, for continuous or highly nonstationary temporal streams, dormancy may prematurely silence neurons that are crucial for processing late-arriving features, potentially causing irreversible information loss and accuracy degradation.

To address the limitations of ITO, the BPTT paradigm incorporates the dormancy mechanism directly into the end-to-end training process. BPTT allows the network to adapt its internal representations to the dormancy constraint, reducing the likelihood that critical neurons enter dormancy prematurely and helping preserve late-arriving features. Through the surrogate-gradient signal, the network learns a robust dormancy policy. In this way, BPTT can help preserve task accuracy across static and dynamic temporal regimes while allowing the network to learn a task-specific balance between computational sparsity and performance.

The inference-time hardware overhead is the same for both the ITO and BPTT paradigms because both use the same dormancy mechanism. The dormancy state is logically implemented as a 1-bit mask per neuron with a small storage footprint, allowing all subsequent 32-bit weight-memory accesses and membrane-potential updates for dormant neurons to be bypassed. A recent neuromorphic hardware design [23] has demonstrated that specialized synaptic processing units can leverage both spike and neuron sparsity to bypass over 98% of invalid computations in workloads such as self-attention. In that design, the reported synaptic energy efficiency is below 0.06 pJ/SOP, corresponding to an energy reduction of over 50% compared with standard SNN implementations.

Table 1: Main results on static and neuromorphic classification benchmarks. The DSN is evaluated under two paradigms: inference-time optimization (ITO) applied to pre-trained SNNs, and direct training with BPTT from scratch. Results show reductions in estimated energy ranging from 40% to over 70%, while maintaining accuracy comparable to or better than that of the corresponding baseline SNNs. Notably, the DSN outperforms state-of-the-art static pruning methods on ImageNet, delivering comparable reported cost reductions and improved accuracy.

Dataset	Model	Architecture	T	GSOPs/ GFLOPs	Energy (mJ)	Acc. (%)
CIFAR10	ANN	ResNet-18	-	0.56	2.55	95.93
	Transformer	2-512	-	1.24	5.72	96.82
	MS-ResNet	ResNet-18	4	0.30	0.28	95.64
	SD-Transformer	2-512	4	0.51	0.46	95.60
	DSN (ITO)	ResNet-18	4	**0.15**	**0.14**	**94.90**
		2-512	4	**0.31**	**0.30**	**94.29**
	DSN (BPTT)	ResNet-18	4	**0.15**	**0.14**	**95.34**
		2-512	4	**0.30**	**0.28**	**95.59**
CIFAR100	ANN	ResNet-18	-	0.56	2.56	78.04
	Transformer	2-512	-	1.24	5.72	81.10
	MS-ResNet	ResNet-18	4	0.32	0.30	77.32
	SD-Transformer	2-512	4	0.63	0.58	78.40
	DSN (ITO)	ResNet-18	4	**0.18**	**0.16**	**76.28**
		2-512	4	**0.38**	**0.35**	**77.39**
	DSN (BPTT)	ResNet-18	4	**0.17**	**0.16**	**76.85**
		2-512	4	**0.36**	**0.33**	**79.36**
DVS-Gesture	SD-Transformer	2-256	16	1.98	1.82	96.88
	DSN (BPTT)	2-256	16	**1.13**	**1.05**	**96.88**
DVS-CIFAR10	SD-Transformer	2-256	16	1.67	1.51	78.1
	DSN (BPTT)	2-256	16	**0.80**	**0.72**	**77.9**
ImageNet	ANN [29]	ResNet-18 / 34	-	1.81 / 3.66	8.34 / 16.85	69.76 / 73.31
	Transformer [30]	8-512	-	8.33	38.34	80.80
	MS-ResNet [31]	ResNet-18 / 34	4	-	4.29 / 5.11	63.10 / 69.42
	SEW-ResNet [32]	ResNet-34	4	3.88	4.04	67.04
	Spikformer [33]	8-384	4	6.82	12.43	70.24
	SD-Transformer [34]	8-384	4	-	3.90	72.28
	AT-LIF [35]	ResNet-18	4	$\downarrow\sim 77\%$	-	59.61
	DPAP [36]	ResNet-18	-	$\downarrow\sim 51.7\%$	-	60.41
	Unstru. [21]	ResNet-18	4	$\downarrow\sim 77\%$	-	61.89
	UniSparse [37]	ResNet-18	4	$\downarrow\sim 78\%$	-	61.89
	SEENN [38]	ResNet-34	4	$\downarrow\sim 55.3\%$	-	67.48
	DSN (ITO)	ResNet-18 / 34	4	**0.88 / 1.50**	**1.15 / 1.72**	**62.04 / 67.67**
	DSN (BPTT)	ResNet-18 / 34	4	**0.93 / 1.67**	**1.17 / 1.83**	**64.44 / 69.63**
		8-384	4	**1.62**	**1.65**	**72.12**

4 Experiments

4.1 Experimental Setup

The effectiveness of the proposed DSN model was evaluated across multiple benchmarks. Static image classification performance was assessed using CIFAR10 and CIFAR100 [39], and ImageNet [40]. Neuromorphic classification performance was evaluated on DVS-CIFAR10 [41] and DVS-Gesture [42]. The experiments used two spike-driven SNN architectures, MS-ResNet and the Spike-Driven Transformer, implemented using the SpikingJelly framework [43]. Detailed hyperparameter configurations and experimental settings are provided in the Appendix C.2 and table 5. The reported energy values are analytical estimates based on the hardware model described in Appendix B, actual device-level energy and latency depend on the target hardware.

4.2 Performance Evaluation

Performance on Static Image Datasets. The results in table 1 show that the DSN reduces estimated energy on static image benchmarks. A central result is that the DSN consistently reduces

computational costs by substantial margins, typically exceeding 40% and frequently reaching 50% to 70%, while maintaining accuracy levels comparable to the baseline SNN models. For instance, the DSN (BPTT) applied to ResNet-18 achieves 64.44% on ImageNet with an estimated energy cost of 1.17 mJ. This corresponds to an estimated 73% energy reduction relative to the 4.29 mJ baseline, accompanied by a 1.34% improvement in accuracy. The proposed method also outperforms recent state-of-the-art static pruning methods on ImageNet. At comparable reported cost reductions, the listed ResNet-18 static methods show lower accuracy than the corresponding baseline, whereas the DSN achieves comparable estimated energy savings while improving upon the baseline accuracy. These results support the effectiveness of instance-level dynamic sparsity.

Comparing the two DSN application paradigms illustrates their respective characteristics. The ITO paradigm provides a training-free, plug-and-play optimization method. It delivers substantial efficiency gains across all tested models but typically incurs a modest accuracy penalty. In contrast, the direct training paradigm requires a full training process but generally preserves or improves accuracy relative to the baseline. The results also suggest an interaction between the DSN mechanism and transformer architectures. Applying the DSN to the SD-Transformer consistently yields substantial energy savings with limited impact on accuracy, including an accuracy increase of nearly 1% on CIFAR100. This suggests that the internal dynamics of transformer models may provide favorable conditions for the dormancy mechanism to remove redundant computations.

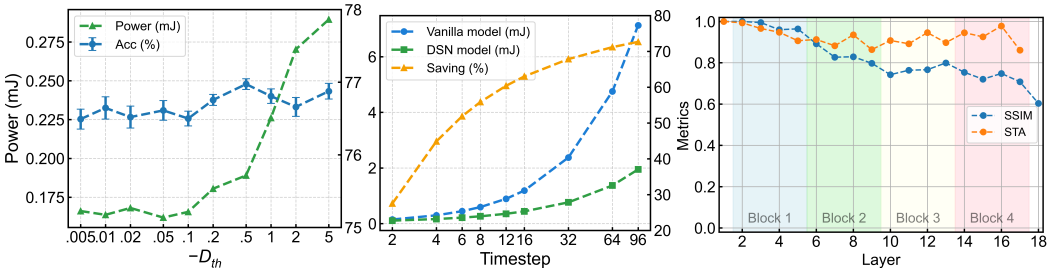
Performance on Neuromorphic Datasets. The DSN was also evaluated on event-based data using DVS-Gesture and DVS-CIFAR10. On DVS-Gesture, the DSN (BPTT) achieves the same 96.88% accuracy as the baseline while consuming 42% less estimated energy. The efficiency improvement is particularly pronounced on DVS-CIFAR10, where the DSN achieves estimated energy savings of over 52% with a 0.2% drop in accuracy. To examine the BPTT paradigm under longer temporal dynamics, we extended the simulation length to $T = 24$ on DVS-CIFAR10. Under this setting, the accuracy deviation from the baseline was 0.4 percentage points. These results suggest that BPTT training can help preserve information from later timesteps over longer temporal horizons.

Training Overhead. In a separate measurement, we quantified the training overhead of the DSN-BPTT by measuring training throughput and GPU memory usage for ResNet-18 on ImageNet using a single GPU with a batch size of 128. The baseline SNN and DSN-BPTT used the same backbone architecture and training settings. The baseline SNN achieved 742.4 images/s and used 15.3 GB of GPU memory, whereas DSN-BPTT achieved 601.6 images/s and used 20.5 GB. These values correspond to a 19% reduction in training throughput and a 34% increase in GPU memory usage. The DSN achieves reductions in estimated inference energy ranging from 40% to over 70%, providing a favorable trade-off for the intended deployment setting.

4.3 Ablation Studies and Analysis

Analysis of Dormancy Threshold. The dormancy threshold D_{th} controls the trade-off between the aggressiveness of computational pruning and model performance. To investigate this relationship, we evaluated ResNet-18 on CIFAR100 across three random seeds to assess run-to-run reliability. The results, plotted in fig. 3a, reveal two primary trends. First, accuracy remains relatively stable, with small standard deviations across the tested range of D_{th}. A slight upward trend in accuracy is observable as the threshold becomes more negative, corresponding to more conservative pruning. Second, energy consumption exhibits a distinct pattern: it remains low and relatively constant in the aggressive pruning region (where $|D_{th}| \leq 0.1$) before rising sharply as the threshold becomes more conservative. This indicates diminishing returns for energy savings when making the pruning strategy less aggressive than $D_{th} \approx -0.1$. Based on this analysis, $D_{th} = -0.05$ was selected for static datasets, providing substantial energy savings near the low-energy plateau while preserving accuracy in the tested range. For neuromorphic datasets, a more conservative threshold of $D_{th} = -0.5$ was adopted to prioritize performance stability under dynamic event-based data streams.

Multi-seed Evaluation. To assess run-to-run reliability, we repeated the BPTT-trained DSN configurations corresponding to the main results using three random seeds. CIFAR10, CIFAR100, and ImageNet use ResNet-18, whereas DVS-Gesture and DVS-CIFAR10 use the Spike-Driven Transformer. The mean and standard deviation of the classification accuracy are reported in table 2. The standard deviations remain small across all evaluated datasets, indicating limited run-to-run variation.



(a) Dormancy threshold D_{th}. (b) Temporal dynamics. (c) Impact of ITO.

Figure 3: Ablation studies on DSN dynamics. (a) Impact of the dormancy threshold D_{th} on accuracy and energy savings evaluated across three random seeds. (b) Energy savings of the DSN compared with the baseline SNN as a function of the number of simulation timesteps T. (c) ITO on network representations. Layer-wise comparison of the SSIM and spike train accuracy (STA) metrics between a pre-trained baseline ResNet-18 SNN and the same network modified using the DSN ITO paradigm.

Table 2: Three-seed accuracy results for the BPTT-trained DSN. Values are reported as mean $\pm$ standard deviation over three runs.

	CIFAR10	CIFAR100	ImageNet	DVS-Gesture	DVS-CIFAR10
Accuracy (%)	95.30 ± 0.10	76.61 ± 0.13	64.36 ± 0.19	96.65 ± 0.40	77.83 ± 0.12

Impact of Simulation Timesteps on Energy Savings. The efficiency benefit provided by the DSN increases with the number of timesteps, as more neurons are progressively identified as deeply inhibited and transition to the dormant state. This benefit was evaluated by comparing the energy consumption of a DSN-equipped ResNet-18 with that of its baseline SNN on CIFAR100 over varying simulation lengths T. The results are plotted in fig. 3b. Even with a short simulation window of $T = 2$, the DSN achieves energy savings of approximately 30%. At $T = 4$, it achieves energy savings of 45%. These results indicate that the DSN can provide computational reductions in low-latency regimes and may be compatible with early-exit or dynamic-timestep strategies. Furthermore, this efficiency gap widens considerably as the simulation duration increases, growing to over 63% at $T = 16$ and exceeding 73% by $T = 96$. This demonstrates that the dormancy mechanism becomes increasingly effective over longer sequences.

Impact of ITO on Network Representations. To further understand the mechanisms underlying the effectiveness of the ITO paradigm, its impact on the internal representations learned by the network was analyzed. Two metrics were employed for this layer-wise comparison using a ResNet-18 SNN pretrained on ImageNet. First, the structural similarity index measure (SSIM) [44] was used to assess the similarity between the pre-activation feature maps of the original SNN and the DSN-modified network. High SSIM values indicate that the core spatial features extracted by the layers are well preserved after applying ITO. Second, spike train accuracy (STA) was used to measure the fidelity of the output spike trains generated by each layer after the ITO modification compared to the original SNN. This metric is calculated as the average fraction of exact spike matches over all neurons and timesteps between the original (S) and DSN-modified ($\hat{S}$) spike trains for a given layer L:

$$\text{STA}_L = \frac{1}{N_L \cdot T} \sum_{i=1}^{N_L} \sum_{t=1}^{T} \mathbb{I}(S_{L,i}^t = \hat{S}_{L,i}^t) \tag{7}$$

where N_L is the number of neurons in layer L, and $\mathbb{I}(\cdot)$ is the indicator function. Figure 3c presents the layer-wise SSIM and STA curves. The consistently high STA values indicate that the dormancy mechanism has little effect on the spike-firing patterns of the original network, supporting the claimed orthogonality between state-driven and input-driven sparsity mechanisms.

Analysis of Neuronal Computational Utilization. To provide deeper mechanistic insight into the efficiency gains afforded by the DSN, another metric was introduced to quantify the computational state of neurons. Let $\overline{R}$ denote the average firing rate, and let $\overline{D}$ denote the average fraction of non-dormant neurons. Building upon these quantities, the neuronal computational utilization (NCU)

metric measures the fraction of neurons that either fire or have entered the dormant state:

$$\text{NCU} = \overline{R} + (1 - \overline{D}) \tag{8}$$

An ideal NCU of 1 signifies that no computational resources are spent on neurons lingering in the subthreshold active state (neither firing nor dormant). Recalling the initial motivation, which identified approximately 44% of neurons residing in a deeply inhibited, computationally redundant state, the NCU metric quantifies how much of this computational redundancy is addressed by the DSN. Layer-wise analysis of a BPTT-trained model on ImageNet supports this interpretation. The empirical results suggest that the DSN does not achieve computational efficiency by indiscriminately suppressing the active firing rate $\overline{R}$. Instead, the energy savings are primarily associated with an increased dormancy fraction $(1 - \overline{D})$ among computationally redundant neurons. Across most network layers, the NCU surpasses 70%. This indicates that the DSN converts a large portion of the identified deeply inhibited population into the dormant state, where subsequent computations are bypassed, thereby addressing the computational redundancy identified in the introduction. The remaining NCU gap largely corresponds to neurons in the subthreshold active region, whose optimization falls outside the scope of this specific dormancy mechanism but suggests avenues for future research.

Jaccard Similarity of Dormancy Masks. To distinguish the DSN from conventional static pruning techniques, the Jaccard similarity of dormant masks across diverse input samples was analyzed. Evaluating 10,000 ImageNet image pairs on a ResNet-34 model revealed high heterogeneity in the dormancy patterns, with an average mask overlap of 0.40. Furthermore, more than 12 layers showed minimal overlap below 0.20. In contrast, for static pruning methods, the Jaccard similarity is fixed at 1 across input samples. This variation supports the conclusion that the DSN leverages instance-level rather than dataset-level redundancy and is consistent with its advantage over static pruning methods.

Validation of the Surrogate Gradient for Dormancy. While applying surrogate gradients is standard practice for handling the non-differentiable spike-firing mechanism in conventional SNN training, the DSN introduces an additional non-differentiable component via the dormancy gate. To validate the effectiveness of the proposed BPTT methodology specifically tailored for the DSN, an ablation study was conducted to compare the use of a smooth SG for the dormancy gate update against a simpler straight-through estimator (STE) [45]. The results show higher accuracy for the SG-based dormancy gate on the evaluated benchmarks. For instance, on the CIFAR10 and CIFAR100 datasets, the SG approach improves accuracy by 0.30% and 0.63%, respectively. More notably, on the large-scale ImageNet dataset, the SG approach achieves 64.44% accuracy, exceeding the STE baseline by 1.57%. This suggests that a smooth gradient signal for the dormancy gate helps the network learn a robust dormancy policy during end-to-end training.

5 Limitations and Discussion

The proposed DSN model demonstrates potential for improving SNN efficiency by directly targeting the computational overhead associated with persistently inhibited neurons. However, the applicability of the DSN framework depends on the neural coding scheme. Specifically, the dormancy mechanism relies on information redundancy and is particularly compatible with rate coding and multi-spike architectures. Conversely, it may be less suitable for low-redundancy coding paradigms, such as Time-to-First-Spike (TTFS) coding. In TTFS schemes, information is encoded in the precise timing of a single spike, leaving limited informational redundancy for the dormancy gate to exploit without risking performance degradation. This limitation does not preclude the practical utility of the proposed method. Networks using TTFS coding generally operate with high sparsity and low baseline computational overhead. Consequently, both the available margin and the practical need for further computational reduction in such highly sparse regimes may be limited.

Beyond efficiency gains, this work suggests a direction for SNN optimization. By shifting the focus from presynaptic input sparsity to postsynaptic state-driven sparsity, the DSN provides a complementary framework for exploiting redundancy in continuous neuronal state updates. Furthermore, insights from the neuronal computational utilization analysis suggest an avenue for future research. While the DSN bypasses computations for deeply inhibited neurons, developing fine-grained optimization strategies for those remaining in the subthreshold active region could improve computational utilization and help explore improved energy-accuracy trade-offs in neuromorphic systems.

6 Conclusion

This work addressed the substantial but often overlooked computational overhead in SNNs arising from continuous subthreshold updates of deeply inhibited neurons. A quantitative analysis characterized the magnitude of this inefficiency, motivating the development of the Dormant Spiking Neuron (DSN). The DSN integrates a mechanism that allows neurons to enter a permanent dormant state based on their membrane potentials, thereby bypassing redundant computations at the instance level. To improve robustness on temporal streams, this state-driven mechanism is trained end-to-end via backpropagation through time using surrogate gradients. Extensive evaluations demonstrated that the DSN framework reduces estimated energy consumption by 40% to over 70% across static and neuromorphic datasets and network architectures. These efficiency gains were achieved while maintaining competitive task accuracy relative to the baselines. On ImageNet, the DSN outperformed the evaluated static pruning methods at comparable reported cost reductions. By providing a trainable, state-driven sparsity mechanism orthogonal to conventional presynaptic input-driven sparsity, the DSN offers an effective approach for improving neuromorphic computing efficiency.

Acknowledgements

This work was supported in part by the Brain Science and Brain-like Intelligence Technology-National Science and Technology Major Project 2022ZD0209700 and in part by Sichuan Provincial Industrial Development Fund under Grant 2025JB04 and in part by Shenzhen Natural Science Foundation under Grant JCYJ20250604180428038.

References

[1] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787–2805, 2010.

[2] Jiasi Chen and Xukan Ran. Deep learning with edge computing: A review. *Proceedings of the IEEE*, 107(8):1655–1674, 2019.

[3] Md. Maruf Hossain Shuvo, Syed Kamrul Islam, Jianlin Cheng, and Bashir I. Morshed. Efficient acceleration of deep learning inference on resource-constrained edge devices: A review. *Proceedings of the IEEE*, 111(1):42–91, 2023.

[4] Ons Aouedi, Thai-Hoc Vu, Alessio Sacco, Dinh C. Nguyen, Kandaraj Piamrat, Guido Marchetto, and Quoc-Viet Pham. A survey on intelligent internet of things: Applications, security, privacy, and future directions. *IEEE Communications Surveys & Tutorials*, 27(2):1238–1292, 2025.

[5] Ruichen Ma, Guanchao Qiao, Yian Liu, Liwei Meng, Ning Ning, Yang Liu, and Shaogang Hu. A&b bnn: Add&bit-operation-only hardware-friendly binary neural network. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5704–5713, 2024.

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

[7] Michael Pfeiffer and Thomas Pfeil. Deep learning with spiking neurons: Opportunities and challenges. *Frontiers in Neuroscience*, Volume 12 - 2018, 2018.

[8] Qi Xu, Yu Qi, Hang Yu, Jiangrong Shen, Huajin Tang, and Gang Pan. Csnn: An augmented spiking based framework with perceptron-inception. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 1646–1652. International Joint Conferences on Artificial Intelligence Organization, 7 2018.

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

[10] Amirhossein Tavanaei, Masoud Ghodrati, Saeed Reza Kheradpisheh, Timothée Masquelier, and Anthony Maida. Deep learning in spiking neural networks. *Neural Networks*, 111:47–63, 2019.

[11] Friedemann Zenke, Sander M Bohté, Claudia Clopath, Iulia M Comşa, Julian Göltz, Wolfgang Maass, Timothée Masquelier, Richard Naud, Emre O Neftci, Mihai A Petrovici, et al. Visualizing a joint future of neuroscience and neuromorphic engineering. *Neuron*, 109(4):571–575, 2021.

[12] Shikuang Deng, Yuhang Wu, Kangrui Du, and Shi Gu. Spiking token mixer: An event-driven friendly former structure for spiking neural networks. In *Advances in Neural Information Processing Systems*, volume 37, pages 128825–128846. Curran Associates, Inc., 2024.

[13] Ruichen Ma, Liwei Meng, Guanchao Qiao, Ning Ning, Yang Liu, and Shaogang Hu. I2e: Real-time image-to-event conversion for high-performance spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 1982–1990, 2026.

[14] Ruichen Ma, Xiaoyang Zhang, Jian Bai, Guanchao Qiao, Liwei Meng, Ning Ning, Yang Liu, and Shaogang Hu. Intrinsically stable spiking neural networks: Overcoming the performance barrier in the absence of batch normalization. In *Computer Vision – ECCV 2026*, pages 507–524, Cham, 2026. Springer Nature Switzerland.

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

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

[17] Ning Qiao, Hesham Mostafa, Federico Corradi, Marc Osswald, Fabio Stefanini, Dora Sumislawska, and Giacomo Indiveri. A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses. *Frontiers in Neuroscience*, Volume 9 - 2015, 2015.

[18] 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.

[19] Nabil Imam and Thomas A Cleland. Rapid online learning and robust recall in a neuromorphic olfactory circuit. *Nature machine intelligence*, 2(3):181–191, 2020.

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

[21] Xinyu Shi, Jianhao Ding, Zecheng Hao, and Zhaofei Yu. Towards energy efficient spiking neural networks: An unstructured pruning framework. In *International Conference on Learning Representations*, volume 2024, pages 50818–50838, 2024.

[22] Yaxin Li, Qi Xu, Jiangrong Shen, Hongming Xu, Long Chen, and Gang Pan. Towards efficient deep spiking neural networks construction with spiking activity based pruning. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 29063–29073. PMLR, 21–27 Jul 2024.

[23] P. J. Zhou, Y. C. Chen, R. C. Ma, G. C. Qiao, N. Ning, Q. Yu, and S. G. Hu. An energy-efficient neuromorphic self-attention core exploiting dual sparsity in neurons and spikes. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 34(1):322–326, 2026.

[24] Yanqi Chen, Zhaofei Yu, Wei Fang, Tiejun Huang, and Yonghong Tian. Pruning of deep spiking neural networks through gradient rewiring. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 1713–1721. International Joint Conferences on Artificial Intelligence Organization, 8 2021. Main Track.

[25] Sayeed Shafayet Chowdhury, Nitin Rathi, and Kaushik Roy. Towards ultra low latency spiking neural networks for vision and sequential tasks using temporal pruning. In *Computer Vision – ECCV 2022*, pages 709–726, Cham, 2022. Springer Nature Switzerland.

[26] Biswadeep Chakraborty, Beomseok Kang, Harshit Kumar, and Saibal Mukhopadhyay. Sparse spiking neural network: Exploiting heterogeneity in timescales for pruning recurrent snn. In *International Conference on Learning Representations*, volume 2024, pages 17995–18026, 2024.

[27] Wenjie Wei, Malu Zhang, Zijian Zhou, Ammar Belatreche, Yimeng Shan, Yu Liang, Honglin Cao, Jieyuan Zhang, and Yang Yang. Qp-snn: Quantized and pruned spiking neural networks. In *International Conference on Learning Representations*, volume 2025, pages 96470–96495, 2025.

[28] 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.

[29] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[30] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[31] 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.

[32] 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, pages 21056–21069. Curran Associates, Inc., 2021.

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

[34] 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, pages 64043–64058. Curran Associates, Inc., 2023.

[35] Tong Bu, Xinyu Shi, and Zhaofei Yu. Activity pruning for efficient spiking neural networks. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 171134–171153. Curran Associates, Inc., 2025.

[36] Bing Han, Feifei Zhao, Yi Zeng, and Guobin Shen. Developmental plasticity-inspired adaptive pruning for deep spiking and artificial neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(1):240–251, 2025.

[37] Xinyu Shi, Tong Bu, and Zhaofei Yu. Unisparse: Combining weight pruning and spike sparsification in spiking neural networks. In *Forty-third International Conference on Machine Learning*, 2026.

[38] Yuhang Li, Tamar Geller, Youngeun Kim, and Priyadarshini Panda. Seenn: Towards temporal spiking early exit neural networks. In *Advances in Neural Information Processing Systems*, volume 36, pages 63327–63342. Curran Associates, Inc., 2023.

[39] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[40] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[41] Hongmin Li, Hanchao Liu, Xiangyang Ji, Guoqi Li, and Luping Shi. Cifar10-dvs: An event-stream dataset for object classification. *Frontiers in Neuroscience*, Volume 11 - 2017, 2017.

[42] Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo Di Nolfo, Tapan Nayak, Alexander Andreopoulos, Guillaume Garreau, Marcela Mendoza, Jeff Kusnitz, Michael Debole, Steve Esser, Tobi Delbruck, Myron Flickner, and Dharmendra Modha. A low power, fully event-based gesture recognition system. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7388–7397, 2017.

[43] Wei Fang, Yanqi Chen, Jianhao Ding, Zhaofei Yu, Timothée Masquelier, Ding Chen, Liwei Huang, Huihui Zhou, Guoqi Li, and Yonghong Tian. Spikingjelly: An open-source machine learning infrastructure platform for spike-based intelligence. *Science Advances*, 9(40):eadi1480, 2023.

[44] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[45] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

A Detailed BPTT Derivation for the DSN

A.1 Forward Dynamics and Gradient Notations

In this section, we provide the rigorous, step-by-step derivations for the BPTT algorithm tailored specifically to the Dormant Spiking Neuron (DSN). We first reiterate the forward propagation dynamics of the proposed DSN model and establish the notations utilized throughout the backward pass. The dynamics of a DSN in layer n at timestep t are governed by the following four stages:

1. Dormancy Gate Update. The binary activity state D_n^t is dynamically updated based on the membrane potential from the preceding timestep V_n^{t-1} and the predefined dormancy threshold D_{th}:

$$D_n^t = D_n^{t-1} \cdot \Theta(V_n^{t-1} - D_{th}) \tag{9}$$

2. Potential Integration. The neuron integrates incoming spikes S_{n-1}^t from the preceding layer, weighted by $\mathbf{W}_n$, with the decayed membrane potential from the previous timestep:

$$H_n^t = f(V_n^{t-1}) + \mathbf{W}_n S_{n-1}^t \tag{10}$$

3. Gated Spike Firing. The output spike S_n^t is generated only if the integrated potential H_n^t exceeds the firing threshold V_{th} and the neuron remains in the active state ($D_n^t = 1$):

$$S_n^t = D_n^t \cdot \Theta(H_n^t - V_{th}) \tag{11}$$

4. Potential Update & Reset. The membrane potential is subsequently updated for the next timestep, incorporating a hard reset mechanism for neurons that have fired:

$$V_n^t = H_n^t(1 - S_n^t) + V_{\text{reset}} S_n^t \tag{12}$$

Here, $f(V)$ represents the membrane potential decay function (e.g., $f(V) = \tau^{-1} \cdot V$ for a standard LIF neuron model). To formalize the BPTT derivation, we define the gradients of the total loss L with respect to the key network variables at layer n and timestep t as follows:

- $\delta_{H,n}^t \equiv \frac{\partial L}{\partial H_n^t}$: Gradient with respect to the integrated potential H.
- $\delta_{S,n}^t \equiv \frac{\partial L}{\partial S_n^t}$: Gradient with respect to the spike output S.
- $\delta_{V,n}^t \equiv \frac{\partial L}{\partial V_n^t}$: Gradient with respect to the updated membrane potential V.
- $\delta_{D,n}^t \equiv \frac{\partial L}{\partial D_n^t}$: Gradient with respect to the dormancy gate state D.

Because the Heaviside step function $\Theta(\cdot)$ in eqs. (9) and (11) is non-differentiable, its exact derivative is substituted with a continuous surrogate gradient function $\sigma'(\cdot)$ during the backward pass:

- $\frac{\partial \Theta(x)}{\partial x} := \sigma_d'(x)$ for the dormancy gate update in eq. (9).
- $\frac{\partial \Theta(x)}{\partial x} := \sigma_s'(x)$ for spike generation in eq. (11).

A.2 Backward Pass Gradient Derivation

The ultimate objective of the BPTT process is to compute the gradient of the loss with respect to the network weights, $\frac{\partial L}{\partial \mathbf{W}_n}$. Using the multivariate chain rule, this gradient is accumulated over the total simulation timesteps T, relying crucially on the intermediate gradient $\delta_{H,n}^t$:

$$\frac{\partial L}{\partial \mathbf{W}_n} = \sum_{t=1}^{T} \frac{\partial L}{\partial H_n^t} \frac{\partial H_n^t}{\partial \mathbf{W}_n} = \sum_{t=1}^{T} \delta_{H,n}^t (S_{n-1}^t)^\top \tag{13}$$

The core task is to unroll the recurrence relations for the defined error terms ($\delta_{H,n}^t$, $\delta_{S,n}^t$, $\delta_{V,n}^t$, $\delta_{D,n}^t$) by propagating the error signals backward through both network depth and time.

A.2.1 Step 1: Deriving $\delta_{H,n}^t$

At timestep t, H_n^t independently influences the current spike generation S_n^t (eq. (11)) and the subsequent reset potential V_n^t (eq. (12)). The gradient $\delta_{H,n}^t$ accumulates from these two paths:

$$\delta_{H,n}^t = \frac{\partial L}{\partial H_n^t} = \underbrace{\frac{\partial L}{\partial S_n^t}\frac{\partial S_n^t}{\partial H_n^t}}_{\text{Firing Path}} + \underbrace{\frac{\partial L}{\partial V_n^t}\frac{\partial V_n^t}{\partial H_n^t}}_{\text{Reset Path}}$$

To ensure training stability and mitigate potential gradient anomalies, it is standard practice in SNNs to detach the spike tensor S_n^t within the reset equation (eq. (12)) from the computational graph during the backward pass (treating it as a constant). This widely adopted approximation simplifies the reset path gradient to $\frac{\partial V_n^t}{\partial H_n^t} = 1 - S_n^t$. Consequently, we obtain:

$$\boxed{\delta_{H,n}^t = \delta_{S,n}^t \cdot \left[D_n^t \cdot \sigma_s'(H_n^t - V_{th})\right] + \delta_{V,n}^t \cdot (1 - S_n^t)} \tag{14}$$

A.2.2 Step 2: Deriving $\delta_{S,n}^t$

The spike output S_n^t propagates forward to influence the integrated potential of the next layer H_{n+1}^t (eq. (10)) and locally affects the intra-neuron reset potential V_n^t (eq. (12)).

$$\delta_{S,n}^t = \frac{\partial L}{\partial S_n^t} = \underbrace{\frac{\partial L}{\partial H_{n+1}^t}\frac{\partial H_{n+1}^t}{\partial S_n^t}}_{\text{Spatial Path}} + \underbrace{\frac{\partial L}{\partial V_n^t}\frac{\partial V_n^t}{\partial S_n^t}}_{\text{Reset Path}}$$

As established in Step 1, the reset path dependency is detached, rendering $\frac{\partial V_n^t}{\partial S_n^t} = 0$. Thus, the error gradient flows exclusively backward through the spatial path from the subsequent layer:

$$\boxed{\delta_{S,n}^t = \mathbf{W}_{n+1}^\top \delta_{H,n+1}^t} \tag{15}$$

A.2.3 Step 3: Deriving $\delta_{V,n}^t$

The membrane potential V_n^t explicitly carries the temporal state to the next timestep $t+1$, jointly influencing the future dormancy gate D_n^{t+1} (eq. (9)) and the future integrated potential H_n^{t+1} (eq. (10)).

$$\delta_{V,n}^t = \frac{\partial L}{\partial V_n^t} = \underbrace{\frac{\partial L}{\partial D_n^{t+1}}\frac{\partial D_n^{t+1}}{\partial V_n^t} + \frac{\partial L}{\partial H_n^{t+1}}\frac{\partial H_n^{t+1}}{\partial V_n^t}}_{\text{Temporal Path}}$$

Evaluating these partial derivatives yields the temporal gradient relation:

$$\boxed{\delta_{V,n}^t = \delta_{D,n}^{t+1} \cdot \left[D_n^t \cdot \sigma_d'(V_n^t - D_{th})\right] + \delta_{H,n}^{t+1} \cdot f'(V_n^t)} \tag{16}$$

Specifically, for the LIF neuron defined by $f(V_n^t) = \tau^{-1} \cdot V_n^t$, the derivative is $f'(V_n^t) = \tau^{-1}$. Conversely, for the IF neuron where $f(V_n^t) = V_n^t$, the derivative evaluates to $f'(V_n^t) = 1$.

A.2.4 Step 4: Deriving $\delta_{D,n}^t$

The dormancy gate state D_n^t structurally governs both its own temporal continuation D_n^{t+1} (eq. (9)) and the immediate spatial spike generation S_n^t (eq. (11)).

$$\delta_{D,n}^t = \frac{\partial L}{\partial D_n^t} = \underbrace{\frac{\partial L}{\partial D_n^{t+1}}\frac{\partial D_n^{t+1}}{\partial D_n^t}}_{\text{Temporal Path}} + \underbrace{\frac{\partial L}{\partial S_n^t}\frac{\partial S_n^t}{\partial D_n^t}}_{\text{Firing Path}}$$

Substituting the derivatives, we formalize the gradient for the dormancy mechanism:

$$\boxed{\delta_{D,n}^t = \delta_{D,n}^{t+1} \cdot \Theta(V_n^t - D_{th}) + \delta_{S,n}^t \cdot \Theta(H_n^t - V_{th})} \tag{17}$$

A.3 The Complete BPTT Algorithm

By consolidating the derived recurrence relations, we establish the complete, implementable algorithmic workflow for the DSN's backward pass.

1. **Initialization:**
   - For all layers $n \in \{1, \dots, N\}$, initialize the temporal error terms originating from the hypothetical future step $T+1$ to zero: $\delta_{H,n}^{T+1} = 0$ and $\delta_{D,n}^{T+1} = 0$.
   - For all layers n, initialize the weight gradient accumulators: $\Delta \mathbf{W}_n = \mathbf{0}$.
   - For all timesteps $t \in \{1, \dots, T\}$, compute the initial error gradient at the final output layer N, $\delta_{S,N}^t = \frac{\partial L}{\partial S_N^t}$, derived directly from the task loss function L.
2. **Iterative Gradient Unrolling:** For each timestep t from T down to 1:
   - For each layer n, iterating backward from N down to 1:
     (a) **Calculate Spatial Spike Error $\delta_{S,n}^t$:** If $n < N$, compute via eq. (15).
     (b) **Calculate Dormancy Gate Error $\delta_{D,n}^t$:** Compute via eq. (17), utilizing $\delta_{D,n}^{t+1}$ (from the previous temporal loop) and $\delta_{S,n}^t$.
     (c) **Calculate Membrane Potential Error $\delta_{V,n}^t$:** Compute via eq. (16), utilizing $\delta_{D,n}^{t+1}$ and $\delta_{H,n}^{t+1}$.
     (d) **Calculate Integrated Potential Error $\delta_{H,n}^t$:** Compute via eq. (14), bridging $\delta_{S,n}^t$ and $\delta_{V,n}^t$.
     (e) **Accumulate Weight Gradient:** $\Delta \mathbf{W}_n \leftarrow \Delta \mathbf{W}_n + \delta_{H,n}^t (S_{n-1}^t)^\top$
3. **Final Weight Update:** Upon completion of the loops, the exact aggregated gradient for the optimizer is defined as $\frac{\partial L}{\partial \mathbf{W}_n} = \Delta \mathbf{W}_n$.

This rigorous formulation ensures that the DSN, complete with its dynamic state-driven sparsity gating, can be trained end-to-end using modern automatic differentiation frameworks.

B Computational and Energy Cost Calculation

B.1 FLOPs Calculation for Standard Architectures

The computational cost of conventional ANNs is commonly reported in FLOPs. For consistency with our energy accounting, one MAC is counted as one FLOP.

For a standard convolutional layer parameterized by C_{in} input channels, C_{out} output channels, a spatial kernel size of $K \times K$, and producing an output feature map of dimensions $H_{out} \times W_{out}$, the total number of dense MACs is formulated as: $\text{MACs}_{\text{Conv}} = H_{out} \cdot W_{out} \cdot K \cdot K \cdot C_{in} \cdot C_{out}$.

Similarly, for a standard fully connected (linear) layer bridging N_{in} input dimensions to N_{out} output dimensions, the total MAC count is: $\text{MACs}_{\text{MLP}} = N_{in} \cdot N_{out}$. The aggregate FLOP count for an entire ANN is simply the algebraic sum of the individual MACs across all constituent layers.

B.2 SOPs Calculation for Spiking Architectures

In stark contrast to ANNs, the computational burden of spiking layers is quantified via SOPs. Because spike activations are binary ($S \in \{0, 1\}$), the heavy floating-point multiplication in an SNN is bypassed, leaving only an event-driven conditional accumulation (AC), which is hardware-efficient. Consequently, the total SOP count for a spiking layer n is merely a fraction of its dense ANN counterpart (MACs_n), fundamentally dictated by the layer's temporal sparsity.

Standard SNNs. Traditional SNNs derive their computational efficiency predominantly from presynaptic input-driven sparsity. A synaptic operation is executed strictly when an active input spike is emitted from the preceding layer $n-1$. Thus, the total SOPs for a baseline spiking layer n equate

Table 3: Energy calculation example for a Spike-Driven Transformer architecture.

		Vanilla Spike-Driven Transformer	Spike-Driven Transformer with DSN
SPS	First Conv	$E_{mac} \cdot \text{MACs}_{Conv_1}$	$E_{mac} \cdot \text{MACs}_{Conv_1}$
	Other Conv	$E_{ac} \cdot \text{MACs}_{Conv_n} \cdot \sum_{t=1}^{T} \overline{R_{n-1}^t}$	$E_{ac} \cdot \text{MACs}_{Conv_n} \cdot \sum_{t=1}^{T} (\overline{R_{n-1}^t} \cdot \overline{D_n^t})$
Self-Attention	Q	$E_{ac} \cdot ND^2 \cdot \sum_{t=1}^{T} \overline{R_{in}^t}$	$E_{ac} \cdot ND^2 \cdot \sum_{t=1}^{T} (\overline{R_{in}^t} \cdot \overline{D_Q^t})$
	K	$E_{ac} \cdot ND^2 \cdot \sum_{t=1}^{T} \overline{R_{in}^t}$	$E_{ac} \cdot ND^2 \cdot \sum_{t=1}^{T} (\overline{R_{in}^t} \cdot \overline{D_K^t})$
	V	$E_{ac} \cdot ND^2 \cdot \sum_{t=1}^{T} \overline{R_{in}^t}$	$E_{ac} \cdot ND^2 \cdot \sum_{t=1}^{T} (\overline{R_{in}^t} \cdot \overline{D_V^t})$
	$f(Q,K,V)$	$E_{ac} \cdot ND \cdot \sum_{t=1}^{T} \overline{R_{KV}^t}$	$E_{ac} \cdot ND \cdot \sum_{t=1}^{T} \overline{R_{KV}^t}$
	Linear 0	$E_{ac} \cdot ND^2 \cdot \sum_{t=1}^{T} \overline{R_{QKV}^t}$	$E_{ac} \cdot ND^2 \cdot \sum_{t=1}^{T} (\overline{R_{QKV}^t} \cdot \overline{D_{MLP_0}^t})$
MLP	Linear 1	$E_{ac} \cdot 4ND^2 \cdot \sum_{t=1}^{T} \overline{R_{MLP_0}^t}$	$E_{ac} \cdot 4ND^2 \cdot \sum_{t=1}^{T} (\overline{R_{MLP_0}^t} \cdot \overline{D_{MLP_1}^t})$
	Linear 2	$E_{ac} \cdot 4ND^2 \cdot \sum_{t=1}^{T} \overline{R_{MLP_1}^t}$	$E_{ac} \cdot 4ND^2 \cdot \sum_{t=1}^{T} (\overline{R_{MLP_1}^t} \cdot \overline{D_{MLP_2}^t})$

to its dense MACs scaled by the average temporal firing rate of the input spikes, denoted as $\overline{R_{n-1}^t}$:

$$\text{SOPs}_{\text{SNN},n} = \text{MACs}_n \cdot \sum_{t=1}^{T} \overline{R_{n-1}^t}$$

The DSN Model. The proposed DSN extends this baseline by introducing a second, orthogonal mechanism: state-driven sparsity. Within the DSN framework, an operation at layer n is executed only if two conditions are simultaneously met: an active input spike arrives from layer $n-1$, and the designated receiving postsynaptic neuron in layer n remains in a non-dormant state. By exploiting this joint filtering effect, the SOP count for a DSN layer n is reduced multiplicatively:

$$\text{SOPs}_{\text{DSN},n} = \text{MACs}_n \cdot \sum_{t=1}^{T} (\overline{R_{n-1}^t} \cdot \overline{D_n^t})$$

Here, the two sparsity factors are defined as follows:
$\overline{R_{n-1}^t}$: The presynaptic activity rate, represented by the mean of the binary spike matrix S_{n-1}^t.
$\overline{D_n^t}$: The fraction of active (non-dormant) neurons in the postsynaptic layer n, represented by the mean of the binary dormancy gate matrix D_n^t.

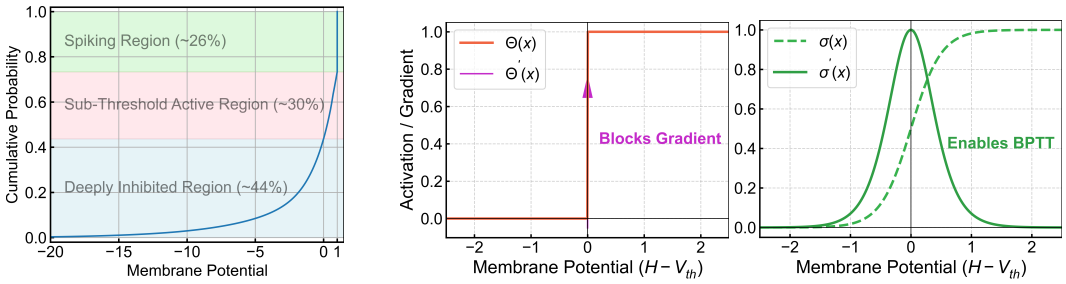
B.3 Energy Consumption Model

To map the aforementioned computational counts to energy estimates, we adopt the standard energy estimation framework established for a 45nm CMOS process node. The theoretical energy costs for 32-bit operations are defined as follows: $E_{mac} = 4.6$ pJ for one dense multiply-and-accumulate (MAC) operation; $E_{ac} = 0.9$ pJ for one sparse accumulate (AC) operation.

The total inference energy for an ANN is computed by integrating the MAC energy costs across all N layers of the network: $\text{Energy}_{\text{ANN}} = E_{mac} \cdot \sum_{n=1}^{N} \text{MACs}_n$.

The energy calculation for spiking architectures is distinct due to the dominance of low-power AC operations. Following standard SNN encoding protocols, the first layer ($n = 1$) operates as an analog-to-spike encoder, executing dense MACs, whereas all subsequent layers ($n > 1$) operate purely in the spike domain, executing sparse ACs. Consequently, the total energy consumption is the hybrid sum of the first layer's MAC energy and the deep layers' SOP (AC) energy:

$$\text{Energy}_{\text{SNN}} = \underbrace{E_{mac} \cdot \text{MACs}_1}_{\text{First Layer (ANN)}} + \underbrace{E_{ac} \cdot \sum_{n=2}^{N} \text{SOPs}_n}_{\text{Spiking Layers}} \tag{18}$$



(a) CDF of membrane potentials. (b) Heaviside step function. (c) Surrogate function.

Figure 4: (a) The cumulative distribution function (CDF) of membrane potentials in a ResNet-18 SNN trained on ImageNet reveals a tripartite split. A substantial fraction (44%) of neurons reside in the deeply inhibited region ($V_m \leq 0$). (b) The Heaviside step function, whose derivative is zero almost everywhere, preventing gradient flow during backpropagation. (c) To enable gradient-based training, the true derivative is replaced by a smooth surrogate function $\sigma'(\cdot)$ during the backward pass.

In this formulation, the term SOPs_n is substituted with $\text{SOPs}_{\text{SNN},n}$ when evaluating a vanilla SNN baseline, and replaced by the highly reduced $\text{SOPs}_{\text{DSN},n}$ when evaluating the proposed DSN model.

B.3.1 Spike-Driven Transformer Energy Analysis

To provide a concrete architectural example, table 3 details the energy calculations for the components of a Spike-Driven Transformer block. This illustrates the precise locations where the DSN injects state-driven sparsity compared to the vanilla SNN baseline. The derivations are formulated based on the sequence length N, the embedding channel dimension D, and the MLP expansion ratio $4D$.

For clarity, the respective spike sparsity rates are defined as follows: $\overline{R_{in}}$ denotes the input spike rate entering the transformer block; $\overline{R_{KV}}$ represents the activation sparsity following the Key-Value (KV) matrix summation; $\overline{R_{QKV}}$ signifies the spike sparsity emerging from the core attention computation $f(Q, K, V)$; and $\overline{R_{MLP_n}}$ denotes the spike rate propagating from the respective linear layers within the MLP block. Correspondingly, the DSN's non-dormant activity factors $\overline{D}$ are applied to the postsynaptic neurons in each internal module, enabling multiplicative reductions in computational cost throughout the self-attention mechanism.

C Supplementary Analysis and Experimental Details

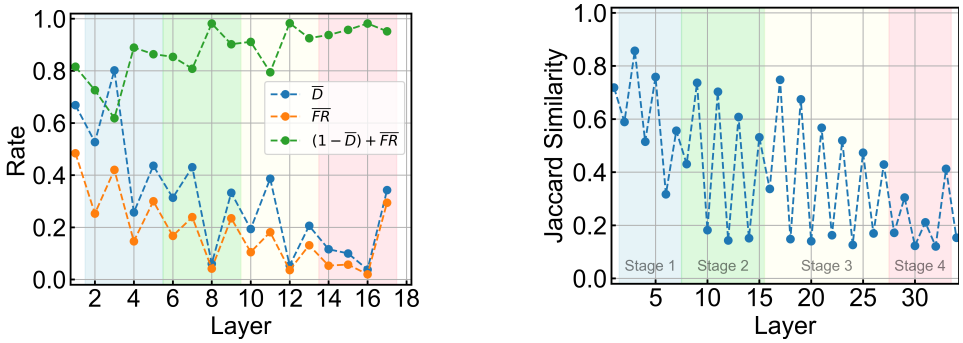
C.1 Supplementary Visualizations and Analysis

Motivation: Membrane Potential Distribution. As posited in the Introduction, the computational inefficiency in standard SNNs primarily stems from the continuous subthreshold state updates of temporarily or permanently inhibited neurons. To quantify this, we tracked the membrane potential distribution of a ResNet-18 SNN model trained on ImageNet. As illustrated by the cumulative distribution function (CDF) in fig. 4a, the internal states of the neurons exhibit a tripartite distribution. While a portion of the neurons resides in the active spiking region or the critical subthreshold active region ($0 < V_m < V_{th}$), 44% of the neuronal population is found in the deeply inhibited region ($V_m \leq 0$). To quantify the recovery probability, we identified neurons whose membrane potentials fell below D_{th} and measured whether they emitted at least one spike during the subsequent timesteps. Averaged across layers and timesteps, approximately 1% of all neurons subsequently fired. These deeply inhibited neurons therefore have a low probability of firing in the immediate future, yet they continue to consume computational resources. This observation motivates the introduction of the dormancy mechanism to dynamically bypass these redundant instance-level computations.

Threshold Sensitivity Across Datasets and Architectures. To further examine the sensitivity of the selected dormancy thresholds, we conducted threshold sweeps with BPTT across multiple datasets and architectures. As reported in table 4, each entry gives the classification accuracy and estimated energy reduction. For the neuromorphic datasets, $D_{th} = -0.5$ provides a favorable performance-

Table 4: Threshold-sweep results across datasets and architectures for BPTT-trained DSN models. Each entry reports accuracy (%) and estimated energy reduction.

D_{th}	DVS-CIFAR10 (SD-Transformer)	DVS-Gesture (SD-Transformer)	ImageNet (ResNet-18)	CIFAR100 (SD-Transformer)
-0.01	75.9% (↓ 71%)	94.44% (↓ 65%)	64.20% (↓ 73%)	79.22% (↓ 44%)
-0.05	76.6% (↓ 70%)	95.48% (↓ 63%)	64.44% (↓ 73%)	79.36% (↓ 43%)
-0.1	77.2% (↓ 68%)	96.18% (↓ 60%)	64.34% (↓ 72%)	79.40% (↓ 42%)
-0.5	77.9% (↓ 52%)	96.88% (↓ 42%)	64.50% (↓ 54%)	79.50% (↓ 32%)
-1	78.0% (↓ 38%)	96.88% (↓ 30%)	64.66% (↓ 37%)	79.55% (↓ 21%)



(a) Firing rate, non-dormant rate, and NCU curves.

(b) Layer-wise Jaccard similarity.

Figure 5: (a) Layer-wise firing rates, non-dormant rates, and the Neuronal Computational Utilization (NCU) on a ResNet-18 architecture. (b) Layer-wise Jaccard similarity of dormancy masks for a ResNet-34 model, showing their heterogeneity across inputs.

efficiency balance. For the static datasets, $D_{th} = -0.05$ remains near the low-energy plateau. These results support the threshold settings used in the main experiments.

Neuron Model and Surrogate Gradient Mechanism. Throughout all experiments, the networks utilize the Leaky Integrate-and-Fire (LIF) neuron model. The intrinsic neuronal parameters were uniformly set across all architectures to ensure stability: the membrane time constant was set to $\tau = 2.0$, the firing threshold to $V_{th} = 1.0$, and the hard reset potential to $V_{\text{reset}} = 0.0$. To circumvent the inherent non-differentiability of the Heaviside step function $\Theta(\cdot)$, which appears in both the spike-firing mechanism and the proposed dormancy gate, we replace its derivative during the backward pass with the continuous surrogate-gradient approximation shown in fig. 4c. Specifically, we derive the surrogate function $\sigma'(x)$ from the sigmoid function:

$$\sigma_d(x) = \sigma_s(x) = \sigma(x) = \frac{1}{1 + e^{-\alpha x}} \tag{19}$$

$$\sigma'_d(x) = \sigma'_s(x) = \sigma'(x) = \frac{\partial \sigma(x)}{\partial x} = \alpha \cdot \sigma(x) \cdot (1 - \sigma(x)) \tag{20}$$

For all BPTT training configurations, the sharpness parameter was empirically set to $\alpha = 4.0$. This smooth approximation provides stable gradient flow for learning robust dormancy policies.

Deep Dive into DSN. To further elucidate the sparsity characteristics of the DSN, fig. 5a details the layer-wise firing rates, non-dormant rates, and the resulting NCU across a ResNet-18 model. The high NCU values across layers indicate that the DSN bypasses computations for a large portion of deeply inhibited neurons without suppressing critical active firing. Furthermore, fig. 5b visualizes the layer-wise Jaccard similarity of the dormancy masks across diverse inputs on a ResNet-34 model. For conventional static pruning techniques, the Jaccard similarity is inherently fixed at 1 across all layers, as the pruned structure remains constant regardless of the input. By contrast, the DSN exhibits substantial mask heterogeneity across layers, with more than 12 layers showing a mean overlap below 0.2. This supports the conclusion that the DSN leverages dynamic, instance-level transient redundancy and adapts its dormancy masks across samples. This capacity for sample-specific dynamic pruning is consistent with the observed advantage over static pruning methods.

Table 5: Training hyperparameters. A summary of the key hyperparameters utilized across all datasets and network architectures.

Dataset	Architecture	Learning Rate	Weight Decay	Batch Size	Epochs	Optimizer
CIFAR	MS-ResNet	2e-2	5e-4	64	300+10	SGD
	SD-Transformer-2-512	3e-4	6e-2	64	300+10	AdamW
ImageNet	MS-ResNet	5e-2	2e-5	64	150+5	SGD
	SD-Transformer-8-384	7.5e-4	1e-2	28	300+10	LAMB
DVS-Gesture	SD-Transformer-2-256	1e-4	6e-2	16	200+10	LAMB
DVS-CIFAR10	SD-Transformer-2-256	2e-3	1e-4	32	200+10	AdamW

C.2 Implementation Details and Training Protocols

Hyperparameters and Data Augmentation. Standard data augmentation pipelines were applied equivalently to all baselines and DSN models to ensure a fair comparison. The comprehensive training hyperparameters, including learning rates, optimizers, and batch sizes, were closely aligned with those used in the original Spike-Driven Transformer to maintain consistency. A detailed summary of these key configurations is provided in table 5.

Training Strategy and Implementation. Under the proposed BPTT paradigm, the DSN models were trained entirely from scratch on the static CIFAR10, CIFAR100, and large-scale ImageNet datasets, as well as the neuromorphic DVS-CIFAR10 dataset. Due to the relatively small size of the DVS-Gesture dataset, the DSN model was fine-tuned from a pre-trained SNN baseline checkpoint to ensure stable convergence. To optimize memory utilization and accelerate the training phase, automatic mixed precision was leveraged across all experiments. For the main experiments and single-seed ablations, we used a fixed random seed of 2025. Multi-seed analyses used the three random seeds described in the main text.

Hardware, Software, and Baselines. All computational experiments were conducted on an internal Linux server running Ubuntu 22.04 with two NVIDIA RTX 5090 GPUs. The software environment used Python 3.10, PyTorch 2.8.0, and SpikingJelly 0.0.0.0.15. For CIFAR10, CIFAR100, DVS-Gesture, and DVS-CIFAR10, baseline results were reproduced internally by running the original, unmodified code under identical hardware and software conditions. Because reproducing ImageNet baselines is computationally costly, the ImageNet baseline results were taken from the published literature for comparison.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the claims regarding the DSN's mechanism, its training via BPTT, and the achieved energy reductions, which match the experimental results and scope presented in the paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We explicitly discuss the limitations of the DSN framework in Section 5, specifically analyzing its compatibility with extreme-latency coding paradigms like TTFS and providing corresponding theoretical justifications.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The full mathematical derivations and chain-rule formulations for the back-propagation through time (BPTT) and surrogate gradients applied to the DSN are provided in the Appendix.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The experimental setup, dataset details, and evaluation metrics are described in Section 4, while comprehensive hyperparameter configurations and training procedures are thoroughly detailed in the Appendix to ensure full reproducibility.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We have committed to making the source code publicly available (as stated in the abstract), and the supplemental material includes an anonymized codebase with detailed instructions to reproduce the main experimental results.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The fundamental experimental setup is presented in the main text, while an exhaustive list of hyperparameter choices, optimizer configurations, and data preprocessing steps are systematically documented in the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report the standard deviation across three random seeds for the critical ablation study concerning the dormancy threshold D_{th} to explicitly demonstrate the robust training convergence of the proposed BPTT method.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Comprehensive information regarding the computational environment, including the type of GPU hardware used for training and inference, is provided in the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This research focuses on foundational algorithmic efficiency for neuromorphic computing and strictly adheres to the NeurIPS Code of Ethics in all aspects.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: This work focuses on foundational mathematical and algorithmic optimizations for energy-efficient neuromorphic systems; it does not have a direct path to specific negative societal applications or harms.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The proposed method is a general algorithmic improvement for SNNs and does not involve the release of foundational models with a high risk for misuse or dual-use.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We properly credit and cite all existing standard datasets (e.g., ImageNet, CIFAR, DVS-Gesture) and open-source frameworks (e.g., SpikingJelly) used in this research.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: The paper introduces a novel algorithm but does not release new datasets or independent pre-trained asset libraries beyond the reproducible codebase itself.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: This research involves purely computational and algorithmic development; it does not involve any crowdsourcing or human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: Since the research does not involve human subjects or biological tissue collection, IRB approval is not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs were not utilized as an active component of the core algorithmic methodology in this research.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.