SPIKING DGCNN: MAX-FIRST GRAPH SPIKING NETWORKS FOR EFFICIENT POINT CLOUD LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Spiking neural networks (SNNs) offer sparse, event-driven computation for energy-efficient learning, but the naive spiking conversion of graph-based point cloud networks remains poorly understood. EdgeConv relies on neighbourhood-wise max aggregation to select discriminative local responses. Simply replacing LeakyReLU with spiking neurons in EdgeConv quantizes features before aggregation, weakening credit assignment and causing severe performance degradation. We therefore propose *Spiking DGCNN*, built on a *Max-First Spiking Rule* that performs neighbourhood selection on continuous-valued features before spike generation. We further introduce an *Adaptive Pseudo-Temporal Expansion-Compression* (APTEC) module that supports lightweight pseudo-temporal evidence accumulation through leaky spiking integration, adaptive thresholding, and binary spike compression. Ablation experiments show that Max-First Spiking Rule improves Spike-before-Max baseline from 79.09% to 91.77% at T=1 and from 84.11% to 92.18% at T=4 on ModelNet40. The complete model with APTEC achieves 92.38% accuracy, closely matching the 92.9% of DGCNN, while reducing the operation-level energy estimate from 11.23 mJ to 0.22 mJ.

1 INTRODUCTION

Point cloud learning is a core problem in 3D vision and underpins a wide range of applications, including object recognition, scene understanding, and embodied perception (Zhao et al., 2021; Wu et al., 2022; Wang et al., 2024). Meanwhile, spiking neural networks (SNNs) have attracted increasing attention as a promising computing paradigm because they employ event-driven computation and sparse binary activations, offering the potential for substantially improved energy efficiency (Roy et al., 2019; Schuman et al., 2022). These properties make SNNs particularly appealing for point cloud analysis, where efficient processing of irregular 3D data is highly desirable, especially in resource-constrained settings such as edge robotics and real-time perception (Qiu et al., 2025). Most point cloud networks extract local geometry by constructing neighbourhoods and aggregating responses from neighbouring points. Following Spiking PointNet (Ren et al., 2023; Lan et al., 2023), recent SNN-based point cloud models have increasingly adopted Transformer-based and state-space-based architectures (Wu et al., 2025a;b). These advances leave a more basic structural question unresolved: *How should neighbourhood operators be combined with spike generation?*

Dynamic Graph CNN (DGCNN) (Wang et al., 2019) is a simple but powerful backbone in artificial neural networks (ANNs) for point cloud learning, known for strong geometric bias and effective local aggregation. A straightforward strategy to construct a spiking DGCNN is to replace the non-spiking activations in EdgeConv with spiking neurons (Zheng et al., 2021; Lee et al., 2016). However, such a naive conversion leads to severe performance degradation. We point out the reason is that EdgeConv relies on neighbourhood-wise max aggregation to preserve the strongest local geometric responses (Wang et al., 2019), whereas spike generation binarizes continuous activations. If binarization is applied before max aggregation, the relative magnitudes among neighbouring responses are collapsed too early, thereby weakening the competition mechanism that makes EdgeConv effective. Consequently, transferring DGCNN into an SNN is not simply a matter of replacing LeakyReLU with spiking neurons (Rueckauer et al., 2017). Rethinking the interaction between spiking dynamics and graph-based local aggregation is necessary.

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

Motivated by this insight, we propose *Spiking DGCNN*, whose key design is a *Max-First Spiking Rule*, in which neighbourhood-wise max aggregation must be performed before spike generation to preserve the neighbourhood competition mechanism in EdgeConv. This rule allows the model to select the strongest local responses in the continuous domain before binary quantization, thereby retaining the geometric discrimination ability of DGCNN under spiking computation. Operation placement addresses geometric selection, but temporal processing introduces another challenge. A real-temporal implementation of spiking DGCNN must repeatedly construct dynamic neighbourhoods and compute edge features across timesteps, substantially increasing the cost of graph processing. We therefore introduce an *Adaptive Pseudo-Temporal Expansion-Compression* (APTEC) module as a lightweight short-window approximation. APTEC reuses graph features over pseudo-timesteps and adaptively accumulates repeated evidence through leaky spiking integration. Experiments on object classification and part segmentation benchmarks show that Spiking DGCNN remains competitive with recent spiking point cloud models (Wu et al., 2025a;b) while providing a favourable estimated energy profile. On semantic segmentation of S3DIS (Armeni et al., 2016), Max-first rules still improves the T=1 mIoU from 38.38% to 48.18%, supporting operation placement as the general mechanism. Although effective on object-centric benchmarks, the complete model with APTEC achieves only approximately 42.23% mIoU on S3DIS at T=4, whereas real-temporal processing with Max-first reaches 50.03%, approaching the 51.51% of DGCNN.

In summary, this work makes the following contributions:

- We revisit DGCNN for spiking point cloud learning and characterize the structural issue in converting neighbourhood-aggregation-based point cloud networks to SNNs.
- We formulate the *Max-First Spiking Rule* to *Spiking DGCNN*, preserving continuous-valued neighbourhood competition before spike generation. We further introduce APTEC as a pseudo-temporal mechanism that avoids repeated dynamic graph computation.
- We validate Max-First across object and scene tasks, and show that *Spiking DGCNN* provides a competitive accuracy–energy trade-off. Our experiments also characterize when pseudo-temporal compression is effective and when real-temporal processing remains necessary.

2 Related Work

2.1 Spiking Neural Networks

SNNs have attracted growing interest as an energy-efficient computing paradigm because of their sparse, event-driven computation and natural compatibility with neuromorphic hardware. Existing training approaches for deep SNNs can be broadly categorized into two classes. The first is ANN-to-SNN conversion (Rueckauer et al., 2017; Jiang et al., 2023; Huang et al., 2025), which transforms a trained ANN into a spiking counterpart. This approach has achieved strong performance and has been extended to support common operations such as max pooling and normalization. However, conversion-based methods often rely on long inference windows and are less effective when the target architecture contains operations that interact poorly with spike quantization (Rueckauer et al., 2017; Deng & Gu, 2021). The second is direct training with surrogate gradients (Lee et al., 2016; Neftci et al., 2019), which optimizes SNNs directly in the temporal domain and offers greater flexibility. Nevertheless, direct training is inherently challenging because spikes are binary, non-differentiable, and temporally dynamic. Recent studies have improved directly trained SNNs through advances in normalization, optimization, and surrogate gradient design, enabling deeper and more effective SNNs (Zheng et al., 2021; Lian et al., 2023).

Despite this progress, most existing advances focused on image classification and related vision tasks. Extending SNNs to point cloud learning is considerably challenging, as point clouds are irregular, unordered, and require explicit local geometry modeling. Our work is related to both conversion-based and directly trained SNNs, but differs in that it focuses on a graph-based point backbone and studies how spike generation interacts with neighbourhood max aggregation in EdgeConv.

2.2 SNN-based Point Cloud Analysis

SNN-based point cloud learning is still in its early stages (Ren et al., 2023; Lan et al., 2023). Spiking PointNet (Ren et al., 2023) introduced the first dedicated SNN framework for point cloud

analysis and demonstrated that point features can be effectively processed in the spiking domain. Building on this foundation, subsequent work has explored more expressive backbones. Spiking Point Transformer (Wu et al., 2025a) extends Transformer-based SNNs to point cloud classification by introducing spiking self-attention together with point-specific encoding strategies. More recently, Efficient Spiking Point Mamba (Wu et al., 2025b) investigates a Mamba-based spiking architecture, enhancing long-range dependency modeling through dedicated temporal mechanisms and asymmetric training strategies.

Overall, existing studies have demonstrated the potential of modern point cloud architectures in the SNNs, while the effective transfer of established ANN designs to SNNs remains underexplored. In this work, we revisit DGCNN and address the challenges of adapting EdgeConv to spiking representations, facilitating the application of proven ANN-based point cloud operators in SNNs.

3 Preliminary and Methodology

3.1 DGCNN and EdgeConv

DGCNN (Wang et al., 2019) models point clouds by explicitly capturing local geometric relations over a neighbourhood graph, rather than processing points independently. Its core operator, ***EdgeConv***, constructs edge features between each point and its neighbours, followed by channel-wise max aggregation. Given a point cloud $\{x_i\}_{i=1}^{N}$ and the neighbourhood $\mathcal{N}(i)$ of point x_i, EdgeConv computes

$$z_{ij} = \phi(x_i, x_j - x_i), \qquad j \in \mathcal{N}(i), \tag{1}$$

where $\phi(\cdot)$ denotes a shared multilayer perceptron (MLP), and $\mathcal{N}(i)$ is defined by a dynamic k-nearest-neighbour graph. The updated feature at point x_i is then given by

$$x_i' = \max_{j \in \mathcal{N}(i)} z_{ij}, \tag{2}$$

where the max is taken channel-wise over neighbouring points. By combining the reference point x_i with the relative offset $x_j - x_i$, EdgeConv captures both local geometric variation and global context. In DGCNN, the neighbourhood graph can be recomputed in feature space at each layer, enabling progressively more semantic neighbourhoods (Wang et al., 2019).

EdgeConv is particularly effective for point cloud learning because it models local geometric interactions, preserves permutation invariance through symmetric aggregation, and extracts discriminative relational features (Wang et al., 2019). However, these properties make DGCNN nontrivial to adapt to a spiking network. A naive ANN-to-SNN pipeline binarizes edge responses before neighbourhood aggregation, weakening the discriminative role of max aggregation (Hu et al., 2023; Wang et al., 2023). The central challenge is therefore to preserve neighbourhood competition before spike quantization.

3.2 Max-First Spiking Rule

The neighbourhood max operator in EdgeConv acts on continuous edge responses and preserves their relative ordering. Introducing spike generation before aggregation distorts this order due to binarization. Motivated by this observation, we study the ordering of *max* and *spike* and adopt a ***Max-First*** spiking rule for EdgeConv. Let $S(\cdot)$ denote a binary spike function,

$$S(x) = H(x - V_{th}), \tag{3}$$

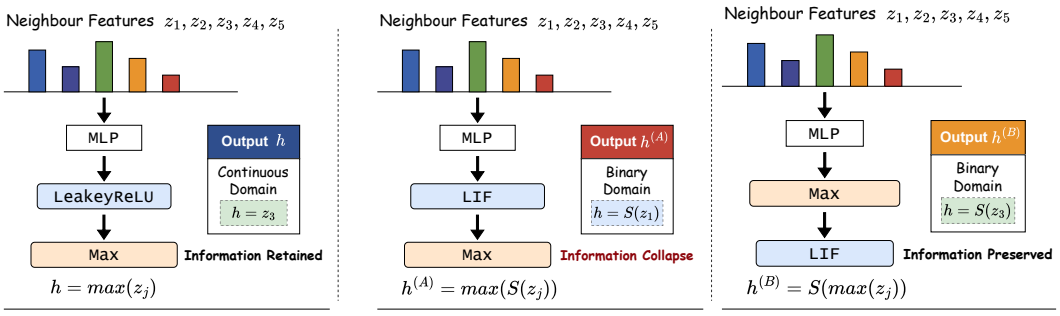
where $H(\cdot)$ is the Heaviside step function and V_{th} is the firing threshold. Two natural spiking variants of EdgeConv are illustrated in Figure 1.

(A) Spike-before-Max:

$$h_i^{(A)} = \max_{j \in \mathcal{N}(i)} S(z_{ij}). \tag{4}$$

Since $S(z_{ij}) \in \{0, 1\}$, the max over neighbors reduces to a logical OR:

$$h_i^{(A)} = \bigvee_{j \in \mathcal{N}(i)} S(z_{ij}). \tag{5}$$



(a) Standard EdgeConv. (b) Spike-before-Max. (c) Max-before-Spike.

Figure 1: Two natural ways of mapping EdgeConv into a spiking network. (a) Standard EdgeConv operates in the continuous domain. (b) Spike-before-Max binarizes responses before aggregation, leading to information collapse. (c) Max-before-Spike preserves the relative ordering of neighbourhood responses, maintaining meaningful competition.

(B) Max-before-Spike (ours):

$$h_i^{(B)} = S\left(\max_{j\in\mathcal{N}(i)} z_{ij}\right). \tag{6}$$

Although these two forms appear similar, they behave differently. The former binarizes features before neighbourhood selection, whereas Max-before-Spike preserves continuous responses until the maximum is selected, as Figure 1. Consequently, they induce different surrogate-gradient routes:

$$\nabla^{\mathrm{SG}}\left[S\left(\max_j z_j\right)\right] \neq \nabla^{\mathrm{SG}}\left[\max_j S(z_j)\right]. \tag{7}$$

where ∇^{SG} denotes the gradient operator under surrogate-gradient-based backpropagation.

Consider a set of neighbourhood responses $\{z_j\}_{j=1}^k \in \mathbb{R}^k$, sorted as

$$z_{(1)} \geq z_{(2)} \geq \cdots \geq z_{(k)}. \tag{8}$$

Under Max-before-Spike, the strongest response is selected before spike generation:

$$h^{(B)} = S(z_{(1)}). \tag{9}$$

Thus, the output depends on the real-valued ordering of the neighbourhood features, preserving the winner-take-all behaviour inherent to EdgeConv. In contrast, under Spike-before-Max,

$$h^{(A)} = \max_j S(z_j) = \begin{cases} 1, & \exists\, j,\ z_j \geq V_{th}, \\ 0, & \text{otherwise}, \end{cases} \tag{10}$$

which is equivalent to

$$h^{(A)} = \mathbb{I}\left(\max_j z_j \geq V_{th}\right). \tag{11}$$

This form only checks whether *any* neighbour exceeds the threshold. All supra-threshold responses are mapped to the same value 1, regardless of their relative magnitudes. As a result, the ordering among neighbours is lost before aggregation.

This difference is also reflected in the computational pathway. Let the neighborhood response be $\mathbf{z} = (z_1, \ldots, z_k) \in \mathbb{R}^k$. Spike-before-Max induces $\mathbb{R}^k \to \{0,1\}^k \to \{0,1\}$, compressing the neighbourhood into a single binary outcome and discarding magnitude relations before aggregation. In contrast, Max-before-Spike induces $\mathbb{R}^k \to \mathbb{R} \to \{0,1\}$, where selection follows the original real-valued ordering. This distinction is critical for EdgeConv, which relies on selecting the strongest local response. If binarization occurs before aggregation, supra-threshold neighbours become indistinguishable, reducing the operator to an activation test rather than a geometric selector.

The two designs also differ in how gradients are propagated during training. For (A) Spike-before-Max $h^{(A)}$, the derivative of $S(\cdot)$ is zero almost everywhere under the hard threshold, so exact

gradients vanish. In practice, surrogate gradients are used. However, even with this approximation, Spike-before-Max remains problematic because once multiple neighbours are binarized to the same value, the max operator cannot distinguish their contributions. As a result, the gradient becomes weak and ambiguous, as several neighbours may share the same binary outcome before aggregation. In contrast, for (B) Max-before-Spike (ours), let

$$j^* = \arg\max_j z_j, \tag{12}$$

so that

$$h^{(B)} = S(z_{j^*}). \tag{13}$$

With surrogate gradients, the learning signal is then routed through the strongest response:

$$\frac{\partial h^{(B)}}{\partial z_{j^*}} \neq 0, \qquad \frac{\partial h^{(B)}}{\partial z_{j \neq j^*}} = 0, \tag{14}$$

where the nonzero term is defined via surrogate gradients. Consequently, Max-before-Spike preserves the winner-take-all mechanism inherent to EdgeConv because it is not structurally ambiguous. Therefore, the model learns which neighbour should dominate the local response, rather than encouraging any neighbour to cross the threshold.

These observations show that the key issue is not whether max aggregation is used, but *where* spike generation is applied relative to neighbourhood selection. Accordingly, we adopt the following Max-First rule throughout our spiking DGCNN:

$$h_i = S\left(\max_{j \in \mathcal{N}(i)} \phi(x_i, x_j - x_i)\right). \tag{15}$$

This rule preserves neighbourhood competition in the continuous domain and enables a faithful spiking adaptation of EdgeConv.

3.3 Pseudo-Temporal Evidence Accumulation with Adaptive Thresholding

Adaptive Pseudo-Temporal Expansion-Compression (APTEC) is an efficient spiking encoding module for point cloud features, which introduces a lightweight temporal dimension enabling spiking neurons to accumulate evidence without the cost of fully time-unrolled graph computation. The module consists of three stages: temporal expansion, adaptive thresholding, and temporal compression.

3.3.1 Temporal Expansion

A key advantage of SNNs lies in their ability to exploit temporal dynamics. However, directly unrolling graph-based backbones such as DGCNN over multiple timesteps is computationally expensive, since graph construction and neighbourhood interactions must be repeated at every step. To avoid this overhead, we construct a *pseudo-temporal* stream by repeating an input feature x across T steps:

$$x \;\rightarrow\; [x^{(1)}, x^{(2)}, \ldots, x^{(T)}], \tag{16}$$

where

$$x^{(1)} = x^{(2)} = \cdots = x^{(T)} = x. \tag{17}$$

Although this process does not generate a true temporal sequence, it introduces a temporal axis along which evidence can be accumulated. The repeated feature stream can then be processed by spiking neurons and subsequently compressed into a compact spike representation.

3.3.2 Adaptive Thresholding

The Limitation of fixed threshold in pseudo-time. If the repeated sequence is processed by a spiking neuron with a fixed firing threshold, the membrane potential evolves as

$$u_t = \lambda u_{t-1} + x^{(t)}, \tag{18}$$

where $\lambda \in [0, 1]$ denotes the decay factor. Because all repeated features are identical, before the first reset, the membrane potential at any step becomes

$$u_t = \left(\sum_{k=0}^{t-1} \lambda^k\right) x = C_t x. \tag{19}$$

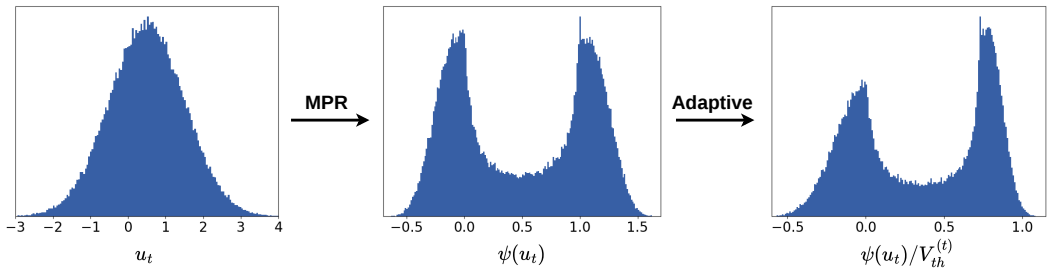


Figure 2: Effect of adaptive thresholding. The membrane potential distribution u_t (left) is first transformed by MPR $\psi(\cdot)$ (middle) and then normalized by the proposed adaptive threshold $V_{th}^{(t)}$ (right). Simulated membrane potentials are sampled from a Gaussian distribution $\mathcal{N}(1/2, 1)$ following (Zheng et al., 2021).

The resulting spike decision is therefore

$$y = H(u_t - V_{th}) = H(C_t x - V_{th}), \tag{20}$$

which is equivalent to applying a binary activation to a scaled input. In other words, increasing T merely amplifies the feature magnitude, causing more units to cross the threshold. As a consequence, temporal compression degenerates into an OR-like behaviour, in which repeated inputs trigger spikes through repeated amplification rather than through meaningful evidence accumulation.

Adaptive thresholding via MPR-inspired decoupling. To reduce spike quantization error, the MPR mechanism redistributes membrane potentials toward binary spike regions before spike generation, while preserving the original membrane state for subsequent temporal dynamics (Guo et al., 2022). Specifically, it decouples the membrane potential used for spike generation from that used for accumulation, thereby improving quantization without altering the true hidden state (Guo et al., 2022). We adopt this decoupling principle in our pseudo-temporal setting, but with a different objective. Beyond reducing quantization error, we introduce an adaptive threshold to suppress spurious activations caused by repeated inputs.

Assume the decision boundary is fixed at 0.5 for the normalized membrane potential. The fire rule is

$$s_t = H\left(\frac{\psi(u_t)}{V_{th}^{(t)}} - 0.5\right), \tag{21}$$

where $\psi(\cdot)$ denotes the Membrane Potential Rectifier (MPR) (Ren et al., 2023). To prevent pseudo-time from collapsing into repeated amplification, we introduce an input-dependent threshold at each timestep:

$$V_{th}^{(t)} = 1 + 0.5 \cdot \sigma(x^{(t)}), \tag{22}$$

where $\sigma(\cdot)$ is the sigmoid function, so that $V_{th}^{(t)} \in (1, 1.5)$. This threshold serves as a *feature-conditioned control* that stronger inputs tend to higher thresholds, thereby introducing negative feedback against over-activation caused by repetition. As a result, adaptive thresholding improves selectivity rather than simply increasing the number of emitted spikes.

Figure 2 illustrates the effect of adaptive thresholding. First, it makes the decision boundary more conservative for marginal units, suppressing spikes induced by repetition. Second, for active units, adaptive normalization shifts the firing variable back into the surrogate-gradient-sensitive regime, improving trainability. Overall, adaptive thresholding both reduces spurious activations in the forward and alleviates gradient saturation during backpropagation. Further analysis is provided in Appendix A.

3.3.3 Temporal Compression

After pseudo-temporal accumulation, APTEC compresses the evidence into a single binary output by aggregating spike responses over time:

$$y = \bigvee_{t=1}^{T} s_t. \tag{23}$$

The output depends on the set of gated spikes $\{s_t\}_{t=1}^{T}$ rather than only the final state u_T. While a single spike is sufficient to activate the output, this is not a trivial OR over repeated inputs. Instead, APTEC follows

$$\text{repeat} \;\rightarrow\; \text{integrate} \;\rightarrow\; \text{adaptive gate} \;\rightarrow\; \text{compress}, \tag{24}$$

in which each s_t is generated from accumulated evidence rather than raw features. This design is well-suited for point cloud learning because pseudo-time provides a space for evidence accumulation, and the final output is a compact binary spike that preserves the sparsity and efficiency of SNNs.

4 EXPERIMENTS

We conduct extensive experiments on four widely used point cloud benchmarks to evaluate the proposed Spiking DGCNN in terms of predictive performance, energy efficiency, and the contribution of its key components. Additional experimental results and details on computational resources are provided in Appendix B and Appendix C. In particular, Appendix B.4 reports semantic segmentation results on S3DIS, evaluating the generalizability of Max-First and APTEC to dense prediction tasks.

4.1 EXPERIMENTAL SETUPS

Datasets. Our experiments are conducted on four widely used point cloud benchmarks, three for classification and one for part segmentation. ModelNet10 (Wu et al., 2015) and ModelNet40 (Wu et al., 2015) are synthetic CAD datasets with 10 and 40 object categories, respectively. They consist of clean and uniformly sampled 3D models and are commonly used for benchmarking classification performance. ScanObjectNN (Uy et al., 2019) is a real-world classification dataset collected from indoor scans, containing 2,902 object instances across 15 categories. Compared with synthetic datasets, it is more challenging due to background clutter, occlusion, and incomplete shapes. Following prior work, we evaluate on three variants: OBJ-BG, OBJ-ONLY, and PB-T50-RS. For part segmentation, we use ShapeNetPart (Yi et al., 2016), which contains 16,881 shapes from 16 categories with a total of 50 part labels. Each category is annotated with 2 to 6 part types, requiring models to predict fine-grained part labels for each point.

Baseline Methods. We compare the proposed Spiking DGCNN with both ANN-based and SNN-based point cloud methods. For ANN baselines, we include representative point-based models, including PointNet (Qi et al., 2017a), PointNet++ (Qi et al., 2017b), and DGCNN (Wang et al., 2019). These methods provide strong non-spiking references for evaluating the performance gap between conventional neural networks and spiking models. For SNN-based baselines, we consider recent representative approaches for point cloud analysis, including Spiking PointNet (Ren et al., 2023), Spiking PointNet++ (Lan et al., 2023), Spiking Point Transformer (SPTransformer) (Wu et al., 2025a), and Efficient Spiking Point Mamba (SPMamba) (Wu et al., 2025b). These methods cover different design paradigms, ranging from point-based architectures to Transformer-based and state-space-based models. All methods are evaluated under their standard settings as reported in the original papers or publicly available implementations. For a fair comparison, we follow common practice and report results on the same benchmarks with consistent evaluation protocols.

Model Architecture & Configuration. We adopt the original DGCNN framework (Wang et al., 2019) for both classification and part segmentation, while replacing its non-spiking activations with the proposed spiking design. Specifically, the EdgeConv operator is reformulated using the proposed Max-First Spiking Rule. For spiking feature encoding, the original LeakyReLU activations are replaced with Leaky Integrate-and-Fire (LIF) neurons (Fang et al., 2021) augmented with MPR (Guo et al., 2022) and the proposed APTEC module. For classification, we follow the standard DGCNN architecture with 1024 input points. For part segmentation, we adopt the segmentation variant of DGCNN with additional feature propagation layers for point-wise prediction, using 2048 input points to better preserve fine geometric details. All models are trained on a k-nearest-neighbour graph ($k = 20$) using stochastic gradient descent (SGD), with the learning rate initialized at 0.1 and decayed to 0.001 via cosine annealing (Loshchilov & Hutter, 2017). Training is conducted for 300 epochs, with batch sizes of 32 for classification and 16 for segmentation. Standard data augmentation, including random scaling and translation, is applied during training.

Table 1: Performance comparison with the baseline methods. We report overall accuracy (%) on ModelNet10, ModelNet40 and ScanObjectNN. The best results in the SNN domain are bolded, with ⋆ indicating self-reproduced results using 1024 points instead of the 2048 points in the original paper.

Arch.	Methods	Params(M)	T	ModelNet10	ModelNet40	ScanObjectNN		
						OBJ-BG	OBJ-ONLY	PB-T50-RS
ANN	PointNet	3.5	-	93.0	89.2	73.3	79.2	68.0
	PointNet++	1.5	-	-	90.7	82.3	84.3	77.9
	DGCNN	1.8	-	95.2	92.9	86.0	86.2	78.1
SNN	Spiking PointNet	3.5	4	93.31	88.61	72.20	76.40	64.10
	Spiking PointNet++	1.5	16	93.54	89.45	-	-	69.22
	SPTransformer	10.2	4	94.76	91.43	82.80	83.40	78.03
	SPMamba	12.8	4	-	92.30	85.71⋆	85.88⋆	77.93⋆
	Spiking DGCNN	1.8	1	94.38	91.94	82.10	84.17	75.57
	Spiking DGCNN	1.8	4	**94.93**	**92.38**	**85.89**	**86.14**	**78.14**

4.2 Results & Discussion

Effectiveness. We evaluate the effectiveness of Spiking DGCNN on both classification and part segmentation benchmarks, as summarized in Table 1 and Table 2. For classification, Spiking DGCNN delivers strong and consistent performance across all datasets. With only 1.8M parameters and a short temporal window T=4, it achieves 92.38% on ModelNet40 and 94.93% on ModelNet10, matching or exceeding existing SNN-based methods. Compared with recent architectures such as SPTransformer and SPMamba, our model attains comparable accuracy with substantially fewer parameters, indicating a more favourable efficiency–performance trade-off. On the more challenging ScanObjectNN benchmark, our method achieves 85.89% on OBJ-BG and 78.14% on PB-T50-RS, outperforming previous SNN-based approaches and approaching the performance of ANN-based DGCNN. For part segmentation, Spiking DGCNN achieves an Ins. mIoU of 85.1%, which is comparable to ANN-based DGCNN (85.2%) and superior to existing SNN-based methods such as SPTransformer and SPMamba. This result shows that the proposed model preserves the fine-grained geometric information required for point-level prediction. Overall, these results show that a simple graph-based spiking model can achieve strong performance without complex architectures. By preserving neighbourhood competition and enabling efficient evidence accumulation, Spiking DGCNN provides an effective and efficient solution for point cloud learning in SNNs.

Table 2: Performance of part segmentation methods. We report Cls. mIoU (%) and Ins. mIoU (%) on ShapeNetPart with T=4.

Architecture	Methods	Cls. mIoU	Ins. mIoU
ANN	PointNet	80.39	83.7
	PointNet++	81.85	85.1
	DGCNN	82.33	85.2
SNN	SPTransformer	81.32	82.9
	SPMamba	**82.29**	84.8
	Spiking DGCNN	82.01	**85.1**

Ablation Study. We further analyze the contribution of each component in Spiking DGCNN, including the Max-First rule, the pseudo-temporal (PT) design, and adaptive thresholding. Results are shown in Table 3. *Effect of Max-First Spiking Rule.* Replacing Spike-before-Max (No.1) with the proposed Max-First rule (No.2) leads to a large performance gain, improving ModelNet40 accuracy from around 79% to over 91% at T=1 and from 84% to over 92% at T=4, with similar trends on other datasets. This confirms that performing spike generation after neighbourhood aggregation is critical for preserving local geometric information. *Effect of Pseudo-Temporal Design.* We then compare real-temporal (RT) and pseudo-temporal (PT) designs without adaptive thresholding (No.2 vs. No.3). PT shows a consistent but small drop compared to RT (e.g., 92.18% to 91.98% on ModelNet40 and 78.83% to 77.83% on PB-T50-RS at T=4), since simple feature repetition mainly amplifies responses and temporal compression degenerates to an OR-like activation. As a result, PT alone does not provide effective evidence accumulation. The T=1 case is omitted since PT and RT are identical in this setting. *Effect of Adaptive Thresholding.* Adding adaptive thresholding to PT significantly improves performance (No.3 vs. No.4). On ModelNet10 and ModelNet40, PT with adaptive thresholding surpasses RT at T=4 (94.93% vs. 94.82%, and 92.38% vs. 92.18%).

Table 3: Ablation Study of Spiking DGCNN on ModelNet10, ModelNet40, and PB-T50-RS. Pseudo-Temporal and Real-Temporal are represented as PT and RT.

No.	Max-first	PT/RT	Adaptive	ModelNet10		ModelNet40		PB-T50-RS	
				T=1	T=4	T=1	T=4	T=1	T=4
1	✗	RT	✗	78.96	91.08	79.09	84.11	60.00	73.00
2	✓	RT	✗	94.14	94.82	91.77	92.18	73.07	78.83
3	✓	PT	✗	—	94.71	—	91.98	—	77.83
4	✓	PT	✓	**94.38**	**94.93**	**91.94**	**92.38**	**75.57**	**78.14**

Table 4: Comparison of different methods in terms of Energy (mJ) and Accuracy on ModelNet40.

	DGCNN	SPTransformer		SPMamba		**Spiking DGCNN**	
TimeStep	ANN	T=1	T=4	T=1	T=4	T=1	T=4
Energy	11.23	3	13.3	1.5	5.4	0.38	**0.22**
Accuracy (%)	92.9	90.87	91.43	91.6	92.3	91.94	**92.38**

This shows that adaptive thresholding effectively prevents over-activation and enables meaningful accumulation. On the more challenging PB-T50-RS, adaptive thresholding improves performance at T=1, while RT remains slightly better at T=4, suggesting real temporal dynamics still help in complex scenarios. Overall, the Max-First Spiking Rule is essential for preserving geometric features, PT alone is insufficient, and adaptive thresholding is key to making Pseudo-Temporal design effective. Together, they enable a simple and efficient spiking model with strong performance.

Energy Cost. We evaluate the energy efficiency of Spiking DGCNN using a standard operation-based estimation protocol (Horowitz, 2014; Zhu et al., 2022; Li et al., 2024). Under a 45nm CMOS technology model, a Multiply-Accumulate (MAC) operation in ANNs consumes 4.6 pJ, while an Accumulate (AC) operation in SNNs consumes 0.9 pJ. Energy is estimated by counting MACs for ANNs and spike-driven AC operations for SNNs. Due to event-driven computation, the effective cost in SNNs is further reduced by spike sparsity. The detailed theoretical energy consumption calculation is provided in Appendix D. As shown in Table 4, Spiking DGCNN achieves large energy savings with competitive accuracy. Compared to ANN-based DGCNN (11.23 mJ), our model requires only 0.38 mJ (T=1) and 0.22 mJ (T=4), corresponding to a 30×–50× reduction with negligible accuracy drop from 92.9% to 92.38%. Compared to other SNN methods, for which increasing timesteps typically increases energy due to repeated computation, our method shows the opposite trend. With APTEC, temporal expansion is followed by compression, so the effective computation remains close to that of a single-timestep design. Meanwhile, the pseudo-temporal process helps optimize spike activity, leading to sparser firing. As a result, increasing T improves accuracy while further reducing energy consumption. This demonstrates that the proposed design achieves an efficient balance between performance and energy.

5 Conclusion

This work investigates how spike generation should be placed relative to neighbourhood selection in spiking point cloud learning. Using EdgeConv, we show that Spike-before-Max and Max-before-Spike preserve different pre-quantization information and gradient routes despite yielding the same hard binary decision. Our *Max-First Spiking Rule* performs neighbourhood selection before spike generation and, when apply to *Spiking DGCNN*, achieves competitive classification and segmentation performance with a favourable accuracy–energy trade-off. More broadly, Max-First provides a potential principle for transferring ANN operators based on max aggregation into the spiking domain. APTEC further reduces training cost by reusing graph features over a short pseudo-temporal window with adaptive thresholding, avoiding fully time-unrolled graph reconstruction. While effective on object-centric tasks, real-temporal processing remains preferable for dense scene segmentation. Future work will extend Max-First to broader max-aggregation architectures and develop task-adaptive temporal schemes that better balance efficiency and temporal dynamics.

AI USE STATEMENT

In accordance with the policy requirements of the ICLR 2027 Conference regarding the use of Large Language Models (LLMs), the authors of this paper hereby declare that: All core academic content of this research—including the formulation of research ideas, design of methodologies, collection and analysis of experimental data, derivation of results, and refinement of conclusions—was independently completed by human authors in its entirety, with no involvement of any LLM tools. LLMs are used only to aid or polish the writing process.

ETHICS STATEMENT

We confirm that all authors have read and comply with the ICLR Code of Ethics. Our work does not involve human subjects, sensitive data collection or release, potentially harmful methodologies or applications, conflicts of interest, discrimination/bias/fairness concerns, privacy/security issues, legal compliance challenges, or research integrity matters that require explicit addressing.

REPRODUCIBILITY STATEMENT

To ensure the reproducibility of our results, we will make our implementation code available as supplementary material. The datasets used (e.g., ModelNet40, ScanObjectNN, ShapeNetPart, S3DIS for point cloud tasks) are publicly accessible, with relevant details provided in the main text. Key experimental settings (including hyperparameters like SNN threshold and timestep optimized in our analysis) and procedures are thoroughly documented in the manuscript. Any theoretical derivations supporting our work are included in the appendix.

REFERENCES

Iro Armeni, Ozan Sener, Amir Zamir, Helen Jiang, Ioannis K. Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1534–1543, 2016.

Shikuang Deng and Shi Gu. Optimal conversion of conventional artificial neural networks to spiking neural networks. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021. URL https://openreview.net/forum?id=FZ1oTwcXchK.

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

Yufei Guo, Yuanpei Chen, Liwen Zhang, YingLei Wang, Xiaode Liu, Xinyi Tong, Yuanyuan Ou, Xuhui Huang, and Zhe Ma. Reducing information loss for spiking neural networks. In *European Conference on Computer Vision*, pp. 36–52. Springer, 2022.

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

Yangfan Hu, Qian Zheng, Xudong Jiang, and Gang Pan. Fast-snn: Fast spiking neural network by converting quantized ann. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45 (12):14546–14562, 2023.

Zihan Huang, Wei Fang, Tong Bu, Peng Xue, Zecheng Hao, Wenxuan Liu, Yuanhong Tang, Zhaofei Yu, and Tiejun Huang. Differential coding for training-free ann-to-snn conversion. In *International Conference on Machine Learning*, pp. 25235–25264. PMLR, 2025.

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

Yuxiang Lan, Yachao Zhang, Xu Ma, Yanyun Qu, and Yun Fu. Efficient converted spiking neural network for 3d and 2d classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9211–9220, 2023.

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

Jintang Li, Huizhe Zhang, Ruofan Wu, Zulun Zhu, Baokun Wang, Changhua Meng, Zibin Zheng, and Liang Chen. A graph is worth 1-bit spikes: When graph contrastive learning meets spiking neural networks. In *12th International Conference on Learning Representations, ICLR*, 2024.

Shuang Lian, Jiangrong Shen, Qianhui Liu, Ziming Wang, Rui Yan, and Huajin Tang. Learnable surrogate gradient for direct training spiking neural networks. In *Ijcai*, pp. 3002–3010, 2023.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017a.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017b.

Xuerui Qiu, Man Yao, Jieyuan Zhang, Yuhong Chou, Ning Qiao, Shibo Zhou, Bo Xu, and Guoqi Li. Efficient 3d recognition with event-driven spike sparse convolution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 20086–20094, 2025.

Dayong Ren, Zhe Ma, Yuanpei Chen, Weihang Peng, Xiaode Liu, Yuhan Zhang, and Yufei Guo. Spiking pointnet: Spiking neural networks for point clouds. *Advances in Neural Information Processing Systems*, 36:41797–41808, 2023.

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

Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in neuroscience*, 11:682, 2017.

Catherine D Schuman, Shruti R Kulkarni, Maryam Parsa, J Parker Mitchell, Prasanna Date, and Bill Kay. Opportunities for neuromorphic computing algorithms and applications. *Nature Computational Science*, 2(1):10–19, 2022.

Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1588–1597, 2019.

Tai Wang, Xiaohan Mao, Chenming Zhu, Runsen Xu, Ruiyuan Lyu, Peisen Li, Xiao Chen, Wenwei Zhang, Kai Chen, Tianfan Xue, Xihui Liu, Cewu Lu, Dahua Lin, and Jiangmiao Pang. Embodiedscan: A holistic multi-modal 3d perception suite towards embodied ai. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19757–19767, June 2024.

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (tog)*, 38(5): 1–12, 2019.

Ziqing Wang, Yuetong Fang, Jiahang Cao, Qiang Zhang, Zhongrui Wang, and Renjing Xu. Masked spiking transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1761–1771, 2023.

Peixi Wu, Bosong Chai, Hebei Li, Menghua Zheng, Yansong Peng, Zeyu Wang, Xuan Nie, Yueyi Zhang, and Xiaoyan Sun. Spiking point transformer for point cloud classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 21563–21571, 2025a.

Peixi Wu, Bosong Chai, Menghua Zheng, Wei Li, Zhangchi Hu, Jie Chen, Zheyu Zhang, Hebei Li, and Xiaoyan Sun. Efficient spiking point mamba for point cloud analysis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 26393–26403, 2025b.

Xiaoyang Wu, Yixing Lao, Li Jiang, Xihui Liu, and Hengshuang Zhao. Point transformer v2: Grouped vector attention and partition-based pooling. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 33330–33342. Curran Associates, Inc., 2022. doi: 10.52202/068431-2415. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/d78ece6613953f46501b958b7bb4582f-Paper-Conference.pdf.

Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, Yuan Xie, and Luping Shi. Direct training for spiking neural networks: Faster, larger, better. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 1311–1318, 2019.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1912–1920, 2015.

Li Yi, Vladimir G Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics (ToG)*, 35(6):1–12, 2016.

Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip H.S. Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 16259–16268, October 2021.

Hanle Zheng, Yujie Wu, Lei Deng, Yifan Hu, and Guoqi Li. Going deeper with directly-trained larger spiking neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 11062–11070, 2021.

Zulun Zhu, Jiaying Peng, Jintang Li, Liang Chen, Qi Yu, and Siqiang Luo. Spiking graph convolutional networks. In Luc De Raedt (ed.), *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pp. 2434–2440. ijcai.org, 2022. doi: 10.24963/IJCAI.2022/338. URL https://doi.org/10.24963/ijcai.2022/338.

Appendix

A Additional Analysis of Adaptive Thresholding

This section provides further analysis of the role of adaptive thresholding in APTEC. The central idea is that the adaptive threshold does not modify the accumulator state itself, but instead acts on the *spike gate* and the *surrogate-gradient-sensitive region*. This property is particularly suitable for pseudo-temporal evidence accumulation, where the main challenge is not insufficient activation, but rather over-activation and saturation caused by repeated inputs. Our design follows the decoupling principle of MPR, in which the original membrane potential is preserved for state evolution, while a modulated membrane potential is used only for spike generation (Guo et al., 2022).

The pseudo-temporal spiking neuron can be written as

$$u_t = \lambda u_{t-1} + x_t, \tag{25}$$

$$\hat{u}_t = \psi(u_t), \tag{26}$$

$$z_t = \frac{\hat{u}_t}{V_{th}^{(t)}}, \qquad V_{th}^{(t)} = 1 + 0.5\,\sigma(x^{(t)}), \tag{27}$$

$$s_t = H(z_t - 0.5), \tag{28}$$

followed by the state update

$$u_t \leftarrow u_t - s_t. \tag{29}$$

Here, u_t denotes the accumulated membrane potential, $\hat{u}_t$ is the rectified membrane potential used for spike generation, and $V_{th}^{(t)}$ is the input-dependent threshold. Only $\hat{u}_t$ and $V_{th}^{(t)}$ participate in spike generation and surrogate backpropagation, whereas the recurrent state evolution is governed by the original membrane potential u_t. Adaptive thresholding therefore acts as a *decoupled gating mechanism*. It controls when spikes are triggered and how gradients propagate, without directly altering the accumulated state.

In our implementation, forward spikes are generated by the hard threshold $H(\cdot)$, while backpropagation is approximated by a surrogate function defined on the normalized variable z_t. Using a clamp-style surrogate, the local gradient is

$$\frac{\partial s_t}{\partial z_t} \approx \mathbf{1}(0 < z_t < 1). \tag{30}$$

By the chain rule, the gradient with respect to the rectified membrane is

$$\frac{\partial s_t}{\partial \hat{u}_t} \approx \frac{1}{V_{th}^{(t)}}\,\mathbf{1}(0 < z_t < 1), \tag{31}$$

and the gradient with respect to the original membrane becomes

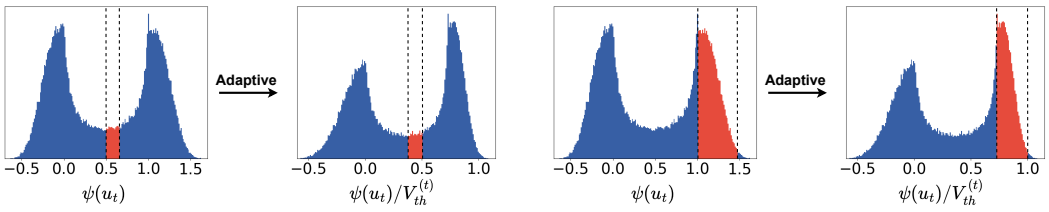
$$\frac{\partial s_t}{\partial u_t} \approx \psi'(u_t)\,\frac{1}{V_{th}^{(t)}}\,\mathbf{1}(0 < z_t < 1). \tag{32}$$

Because the threshold is input-dependent, an additional gradient path arises through the threshold branch:

$$\frac{\partial s_t}{\partial x^{(t)}}\Big|_{\text{th}} \approx -\frac{\hat{u}_t}{\left(V_{th}^{(t)}\right)^2} \cdot 0.5\,\sigma(x^{(t)})\big(1 - \sigma(x^{(t)})\big) \cdot \mathbf{1}(0 < z_t < 1). \tag{33}$$

This expression leads to two important effects. First, the membrane-path gradient is scaled by $1/V_{th}^{(t)}$, making the update more conservative. Second, the model acquires an additional gradient route that adjusts not only the membrane magnitude, but also the neuron excitability under the current input. Thus, adaptive thresholding turns spike generation from a fixed decision rule into a learnable excitability control mechanism.

To further analyze this behaviour, we consider two representative situations of the firing variable z_t. The first corresponds to borderline activations near the decision threshold, where adaptive thresholding suppresses spurious spikes while preserving gradient flow. The second corresponds to saturated activations, where adaptive normalization pulls z_t back into the surrogate-gradient-sensitive region. Together, these two cases illustrate how adaptive thresholding improves both selectivity and trainability.



(a) Case I. (b) Case II.

Figure 3: Effects of adaptive thresholding in two representative cases. Adaptive thresholding reshapes the normalized firing variable $z_t = \psi(u_t)/V_{th}^{(t)}$ in borderline and saturated states.

Case I: borderline pseudo-spikes are suppressed while remaining trainable. Consider a unit whose normalized firing variable under a fixed threshold lies near the decision boundary, e.g.,

$$z_t \in [0.5,\ 0.66]. \tag{34}$$

Under adaptive thresholding, the enlarged denominator may shift it to

$$z_t' \in [0.38,\ 0.5], \tag{35}$$

changing the output from 1 to 0, as illustrated in Figure 3a. This behaviour is desirable in pseudo-time. Without adaptive thresholding, repeated inputs can push such borderline units across the threshold, producing spurious spikes driven by repetition rather than by strong evidence. Once a spike is emitted, a reset is triggered, which may interrupt subsequent accumulation. In contrast, adaptive thresholding delays firing, allowing the membrane potential to be retained for later timesteps.

Importantly, although the forward decision becomes more conservative, these units remain trainable. As long as

$$0 < z_t' < 1, \tag{36}$$

the surrogate gradient remains nonzero. Thus, suppressed borderline activations still contribute to learning and can be pushed back above the threshold if necessary. In this sense, adaptive thresholding does not merely reduce firing. Rather, it makes the forward more selective while preserving effective backpropagation. This is particularly beneficial in pseudo-time, where the goal is to suppress repetition-induced false positives without preventing genuinely informative spikes.

Case II: saturated active spikes are desaturated into the trainable region. The second case, illustrated in Figure 3b, involves active units whose normalized firing variable is excessively large, e.g.,

$$z_t \in [1,\ 1.48]. \tag{37}$$

These units still produce $s_t = 1$, but receive zero gradient under the clamp-style surrogate because they lie outside the active region. Adaptive thresholding can shift them to

$$z_t' \in [0.73,\ 1.0], \tag{38}$$

where the forward decision remains unchanged while the gradient becomes nonzero. This highlights another key benefit that adaptive thresholding not only suppresses undesirable spikes, but also *desaturates* strongly active units back into the surrogate-gradient-sensitive region.

Although this effect does not alter the forward output, it is beneficial for optimization. The model avoids placing excessive mass in the saturation region and instead maintains strong activations while restoring their trainability. In other words, adaptive thresholding encourages a *stable but controllable firing*, rather than uncontrolled over-activation. This aligns with the goal of pseudo-temporal encoding, where spikes reflect accumulated evidence rather than repeated inputs.

B Additional Experimental Results

B.1 Effect of Time Steps in Pseudo-Temporal Encoding

We further study the impact of the number of timesteps T in the proposed pseudo-temporal (PT) encoding, as shown in Table 5. The performance improves steadily from T=1 to T=4, reaching the

Table 5: Accuracy (%) on ModelNet40 under different time steps T.

TimeStep	T=1	T=2	T=3	T=4	T=5	T=6
Accuracy (%)	91.94	92.14	92.22	**92.38**	92.19	91.97

Table 6: Additional Ablation Study on Adaptive Thresholding in Real-Temporal (RT) and Pseudo-Temporal (PT) Settings.

No.	Max-first	PT/RT	Adaptive	ModelNet10		ModelNet40		PB-T50-RS	
				T=1	T=4	T=1	T=4	T=1	T=4
1	✓	RT	✗	94.14	94.82	91.77	92.18	73.07	78.83
2	✓	RT	✓	**94.38**	94.82	**91.94**	92.22	**75.57**	**79.60**
3	✓	PT	✓	—	**94.93**	—	**92.38**	—	78.14

highest accuracy of 92.38%. This result suggests that a moderate temporal length enables the model to accumulate useful evidence and refine spike activity more effectively. However, as T increases further, performance gradually declines (e.g., 92.19% at T=5 and 91.97% at T=6). This observation is consistent with our analysis that excessive repetition can lead to over-amplification and reduced discriminability in PT encoding, even when temporal compression is applied. Unlike real-temporal designs, where longer sequences may provide richer temporal dynamics, PT is still an approximation and suffers from performance degradation when T becomes too large. Overall, PT achieves its best performance with a short temporal window, offering a favourable balance between accuracy and efficiency. This result indicates that the proposed method can capture most of the benefits of temporal processing without requiring long time steps.

B.2 Effect of Adaptive Thresholding in Real-Temporal and Pseudo-Temporal Settings

We further analyze the effect of adaptive thresholding under both real-temporal (RT) and pseudo-temporal (PT) settings, as shown in Table 6. Under the RT setting, introducing adaptive thresholding yields clear performance gains at $T = 1$ (No.1 vs. No.2), indicating that adaptive thresholding is effective even without pseudo-temporal expansion because it improves spike selectivity and stabilizes activation. When adaptive thresholding is used, RT and PT are equivalent at T=1. Therefore, PT results are omitted in this case. At T=4, PT achieves slightly higher accuracy than RT on ModelNet10 and ModelNet40, suggesting that pseudo-temporal accumulation can provide additional benefits when combined with adaptive thresholding. However, on the more challenging PB-T50-RS benchmark, RT still performs better (79.6% vs. 78.14%), indicating that real-temporal dynamics remain advantageous in more complex scenarios.

Overall, adaptive thresholding consistently improves performance in both RT and PT settings. It enhances spike selectivity and mitigates over-activation, serving as a general mechanism for improving spiking representations. While PT benefits from adaptive thresholding on relatively simpler tasks, RT retains an advantage in more challenging cases.

B.3 Layer-wise Firing Rate Analysis

We further analyze the layer-wise firing rates (f_r) and the operation distribution under different object-level settings, as summarized in Table 7. When the temporal length increases to T=4, we observe a consistent reduction in firing rates across the most energy-intensive layers (Conv3 - Conv5) on all datasets. These layers also correspond to the highest MAC/AC costs, indicating that the reduction in firing activity directly contributes to lower overall energy consumption. This behaviour shows that, with adaptive thresholding, increasing T does not lead to naive accumulation or uncontrolled over-activation. Instead, spike activity is progressively suppressed and refined, resulting in sparser and more selective firing patterns. This provides a physical-level explanation for the effectiveness of adaptive thresholding. It regulates neuron excitability and prevents redundant spikes, particularly in deeper and more computationally expensive layers.

Table 7: Layer-wise MACs & ACs (M) and firing rates (fr) under different settings.

Model	T	Conv Layers					FC Layers		
		Conv1	Conv2	Conv3	Conv4	Conv5	Fc1	Fc2	Fc3
MACs & ACs (M)		13.1	173	346	1363.1	541.1	4.2	0.5	0.1
ModelNet10	1	1	0.30	0.22	0.12	0.04	0.74	0.18	0.22
	4	1	0.41	0.18	0.08	0.04	0.82	0.13	0.17
ModelNet40	1	1	0.36	0.23	0.13	0.05	0.69	0.19	0.28
	4	1	0.58	0.12	0.02	0.01	0.82	0.16	0.16
PB-T50-RS	1	1	0.39	0.28	0.15	0.04	0.57	0.19	0.10
	4	1	0.41	0.18	0.04	0.02	0.74	0.17	0.19

Table 8: Semantic Segmentation Results on S3DIS. Pseudo-Temporal and Real-Temporal processing are represented as PT and RT, respectively. PT corresponds to the APTEC setting.

No.	Architecture	Max-first	PT/RT	S3DIS	
				T	mIoU (%)
1	ANN	—	—	—	51.51
2	SNN	✗	—	1	38.38
3	SNN	✓	—	1	48.18
4	SNN	✓	PT	4	42.23
5	SNN	✗	RT	4	42.01
6	SNN	✓	RT	4	**50.03**

The pseudo-temporal mechanism further benefits from this property. By combining temporal expansion-compression with adaptive control, the model achieves better performance at T=4 while simultaneously reducing firing rates in high-cost layers. This explains why our method can obtain higher accuracy with lower energy consumption, highlighting the joint advantage of pseudo-temporal encoding and adaptive thresholding.

B.4 Semantic Segmentation on S3DIS

We further evaluate Spiking DGCNN on S3DIS (Armeni et al., 2016) to examine whether the conclusions obtained from object benchmarks extend to indoor scene segmentation. Compared with object classification and part segmentation, semantic segmentation on S3DIS requires point-wise predictions in cluttered indoor scenes with spatially heterogeneous structures, providing a more challenging setting for evaluating both the generalizability of the Max-First Spiking Rule and the applicability of pseudo-temporal processing. We still use DGCNN (Wang et al., 2019) as the ANN reference and evaluate five spiking configurations as in Table 8. All models follow the same data preprocessing, training split, segmentation backbone, and Area-5 evaluation protocol, except for the operation ordering and temporal setting under investigation. We report mean intersection over union (mIoU) as results. Because pseudo-temporal and real-temporal processing reduce to the same single-step computation at $T = 1$, no temporal setting is assigned to the two $T = 1$ configurations in Table 8.

Effect of Max-First Spiking Rule. Replacing Spike-before-Max with the proposed Max-First rule improves mIoU from 38.38% to 48.18% at T=1. A consistent improvement is observed with real-temporal processing at T=4, where mIoU increases from 42.01% to 50.03%. These results confirm that performing spike generation after neighborhood aggregation is also critical for semantic segmentation, where preserving discriminative local responses is important for fine-grained point-wise prediction. The consistent gains across temporal settings further demonstrate the effectiveness of the Max-First rule for dense scene understanding.

Effect of Temporal Design. We then compare pseudo-temporal (PT) and real-temporal (RT) processing with Max-First at T=4. RT achieves 50.03% mIoU, substantially outperforming PT with APTEC (42.23%) and remaining only 1.48 points below DGCNN (51.51%). This indicates

that APTEC is less effective for complex scene segmentation, where repeatedly reusing a fixed graph representation provides limited temporal information for fine-grained point-wise prediction. In contrast, real-temporal processing enables more effective evidence accumulation and achieves performance comparable to its ANN counterpart.

Overall, these results further confirm the effectiveness of Max-First in preserving fine-grained geometric information, with consistent improvements from object-level recognition to dense scene segmentation. Meanwhile, APTEC provides an efficient alternative to real-temporal processing on object-level benchmarks but is less effective for complex dense scenes such as S3DIS. This suggests that Max-First generalizes well across different point cloud tasks, while the effectiveness of pseudo-temporal processing is more task dependent.

C Experimental Compute Resources

All experiments are conducted using an AMD EPYC 7763 CPU (8 cores) and an NVIDIA RTX A6000 GPU with 48 GB memory. The system is configured with 20 GB RAM and NVMe SSD storage. This configuration is sufficient for training and evaluating all models in our experiments, including both classification and part segmentation tasks.

D Theoretical Energy Consumption Calculation

In the experimental section of our paper, we utilize energy consumption as a metric to evaluate the efficiency of various models. Here, we describe how the theoretical energy of the proposed SNN is estimated. The procedure consists of two steps: estimating the number of effective synaptic operations and converting them into energy cost.

For each layer l of the SNN, the number of synaptic operations (SOPs) is approximated as

$$\mathrm{SOPs}(l) = f_r \cdot T \cdot \mathrm{FLOPs}(l), \tag{39}$$

where f_r denotes the average firing rate of the input spikes to the layer, T is the number of timesteps, and $\mathrm{FLOPs}(l)$ is the number of floating-point operations in the corresponding ANN layer, primarily multiply-accumulate (MAC) operations. This formulation reflects the fact that, in SNNs, only a fraction of synaptic operations are activated because spike firing is sparse.

In our proposed framework, however, the effective temporal cost is modified by the *Adaptive Pseudo-Temporal Expansion-Compression* (APTEC) module. Although the model is trained with a pseudo-temporal length T, APTEC performs temporal expansion followed by compression, such that spike accumulation is internally aggregated into a single output. As a result, the effective number of synaptic operations does not scale with T in the same way as conventional real-temporal SNNs. **From the perspective of energy estimation, this allows us to treat the temporal factor as T=1 when computing SOPs**, since the final spike response is generated after compression rather than through repeated full computations across timesteps. Therefore, while Eq. 39 describes the general case, the practical energy cost in our model corresponds to a compressed temporal execution.

To estimate energy consumption, we consider two types of operations under a 45 nm CMOS technology model: multiply-accumulate (MAC) and accumulate (AC). Their energy costs are set to

$$E_{\mathrm{MAC}} = 4.6\ \mathrm{pJ}, \qquad E_{\mathrm{AC}} = 0.9\ \mathrm{pJ}. \tag{40}$$

In our implementation, the first convolutional layer processes dense inputs and is therefore computed using MAC operations, whereas the remaining layers operate on spike-driven inputs and primarily involve AC operations.

Following the methodology in (Wu et al., 2019; 2025a;b), the total energy consumption of the SNN is computed as

$$E_{\mathrm{SNN}} = E_{\mathrm{MAC}} \cdot \mathrm{FLOPs}_{\mathrm{conv}}^{(1)} + E_{\mathrm{AC}} \cdot \left(\sum_{n=2}^{N} \mathrm{SOPs}_{\mathrm{conv}}^{(n)} + \sum_{m=1}^{M} \mathrm{SOPs}_{\mathrm{fc}}^{(m)} \right), \tag{41}$$

where N and M denote the numbers of convolutional and fully connected layers, respectively. Here, $\mathrm{FLOPs}_{\mathrm{conv}}^{(1)}$ corresponds to the first convolutional layer, while the remaining terms account for spike-driven operations in deeper convolutional and fully connected layers.

For ANN models, all operations are dense MAC computations, and the total energy consumption is estimated as

$$E_{\mathrm{ANN}} = E_{\mathrm{MAC}} \cdot \mathrm{FLOPs}_{\mathrm{ANN}}, \tag{42}$$

where $\mathrm{FLOPs}_{\mathrm{ANN}}$ denotes the total number of floating-point operations of the ANN model.