A SPIKING FRAMEWORK FOR EFFICIENT GRAPH REPRESENTATION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Integrating spiking neural networks (SNNs) into graph neural networks (GNNs) offers a promising approach to energy-efficient graph representation learning. However, existing integration methods can still exhibit performance gaps compared with strong real-valued GNNs, while adapting SNNs to these diverse GNN architectures remains challenging. To bridge this gap, we propose a spiking GNN framework applicable to various message-passing neural networks (MPNNs) and graph Transformers. Specifically, for MPNNs, we integrate spiking neurons between layers and reposition the linear transformation modules to the start of each layer. This structure enables the computation of linear transformation to become sparse addition operations. Moreover, our method exhibits a linear decay rate of Dirichlet energy, effectively alleviating the over-smoothing problem. For graph Transformers, we design a novel spike-based self-attention mechanism and use layer normalization to replace the softmax function, which can better align with the discrete characteristics of SNNs and greatly reduce computational overhead. We extend our method to six MPNN models and three graph Transformer models. Without sacrificing model performance, the energy consumption of spike-based models is approximately 10% of that of non-spiking counterparts.

1 INTRODUCTION

Graph neural networks (GNNs) Scarselli et al. (2008) can be specifically divided into message-passing neural networks (MPNNs) Gilmer et al. (2017) and graph Transformers. For MPNNs, the core characteristic lies in the aggregation of local neighborhood features, such as graph convolutional networks (GCNs) Kipf & Welling (2016), graph attention networks (GAT) Velickovic et al. (2017), and graph isomorphism networks (GIN) Xu et al. (2018). However, MPNNs often suffer from the problems of over-smoothing Li et al. (2018) and weak feature extraction capability. In contrast, graph Transformers Vaswani et al. (2017); Dwivedi & Bresson (2020) can adaptively determine the importance of other nodes to the central node based on the relationships between them, delivering strong performance Yuan et al. (2022); Chen et al. (2022); Buterez et al. (2025). Both MPNNs and graph Transformers rely on continuous-valued activations and dense multiply-accumulate operations (MACs) (see Figure 1 (a)). In particular, graph Transformers need to calculate the attention between every pair of nodes, resulting in a computational complexity of $\mathcal{O}(N^2)$, where N is the number of nodes. With the growth of the graph scale, this computationally intensive property leads to a high energy consumption Li et al. (2022); Rao et al. (2023); Zhuo et al. (2025).

Spiking neural networks (SNNs) Maass (1997) offer a promising direction by communicating through discrete spikes. When sparse spikes serve as inputs to linear transformations, the corresponding computations can be implemented through spike-triggered accumulations, creating opportunities for energy savings under suitable execution conditions Wu et al. (2022); Li et al. (2025); Ding et al. (2025). Existing studies have explored spiking graph convolutional networks Zhu et al. (2022) and spiking graph Transformers Sun et al. (2024b); Qian & Tao (2025). Building on these advances, three challenges motivate our work:

- **Leveraging established GNN architectures.** Strong real-valued GNNs benefit from effective designs in neighborhood aggregation, attention, and structural modeling. However, these designs are not always preserved in existing spiking graph models, which still exhibit

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.

performance gaps compared with strong real-valued GNNs. This motivates us to construct spiking counterparts from established GNN architectures to retain their strengths.

- **Adapting diverse graph architectures.** Graph learning encompasses diverse message-passing schemes and graph Transformer architectures. Designing a separate spiking variant for each model requires substantial architecture-specific effort. Therefore, a framework that can be applied across different GNN architectures is highly desirable.
- **Understanding spiking dynamics and over-smoothing.** Previous work has empirically shown that spiking dynamics can alleviate over-smoothing in deep graph networks Xu et al. (2021), while the underlying theoretical mechanism remains unclear.

To bridge this gap, we propose a universal method to integrate SNNs into various MPNNs and graph Transformers, respectively. For MPNNs, linear layers typically dominate the computational overhead Wu et al. (2019). To address this, we interleave spiking neurons between consecutive layers and uniformly position the linear layers at the beginning of each MPNN block. This configuration ensures that the linear layers receive sparse spike sequences as inputs, so that the most computationally intensive part consists of sparse accumulation operations. Moreover, due to the spike output nature of SNN, similar features around the spike firing threshold will be amplified into distinctly different outputs, effectively mitigating the over-smoothing issue (see Figure 1 (b)). We formally characterize this property through a rigorous theorem (see Theorem 1). For graph Transformer models, we present a spike-based attention computation method by converting queries (Q) and keys (K) into spiking sequences. However, due to the discrete nature of spikes, the resulting attention scores obtained from Q and K are restricted to a narrow interval. Consequently, the softmax function produces a near-uniform distribution, thereby stripping the attention mechanism of its selectivity. Therefore, we discard softmax and instead incorporate a layer normalization module to stabilize the feature distribution. Notably, our method allows for reordering the computation of Q, K, and V, thus achieving a complexity of $\mathcal{O}(N)$. Subsequently, we extend our framework to attention mechanisms equipped with masking and structural bias (see Figure 3), both of which effectively circumvent dense MAC operations.

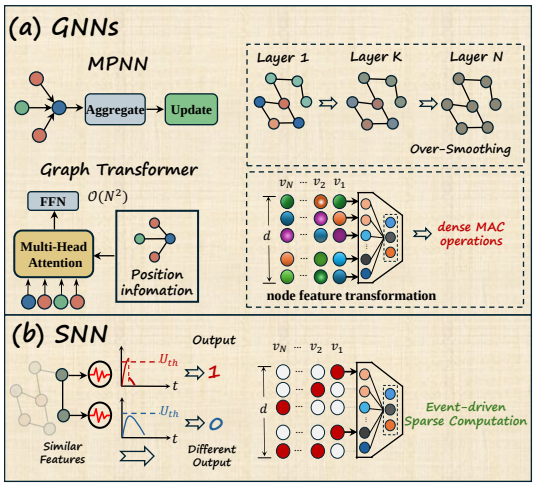

Figure 1: GNNs typically involve dense MAC operations, whereas the spiking characteristics of SNNs transform this process into sparse computation. Besides, for similar inputs, distinct outputs are generated as long as the membrane potential exceeds the firing threshold U_{th}, which effectively alleviates the over-smoothing problem.

We verify our method on six MPNN models and three graph Transformer models. We compare the performance of spiking versions and non-spiking versions of the models. Experimental results demonstrate that the introduction of spiking neurons does not sacrifice the performance of the original models. In terms of energy consumption, we calculate the theoretical energy consumption of each model based on the number of multiplication and addition operations. The energy consumption of spiking versions is approximately 10% of that of non-spiking versions. Furthermore, we experimentally demonstrate that SNNs can effectively alleviate the over-smoothing problem in GNNs.

2 Related Work

2.1 Graph Neural Networks

The distinctions between MPNN models primarily stem from their various methods of node aggregation and update. For example, GCN Kipf & Welling (2016) performs aggregation with a fixed symmetric normalized adjacency matrix. GAT Velickovic et al. (2017) introduces an attention

mechanism that adaptively assigns customized weights to neighboring nodes. GIN Xu et al. (2018) employs the multi-layer perceptron (MLP) to approximate invertible functions for strong discriminative power. However, these models face an over-smoothing bottleneck as network depth increases Li et al. (2018). Graph Transformers Dwivedi & Bresson (2020) extend graph representation learning with attention-based interactions and explicitly incorporate graph structural information. Representative designs differ in both the scope of node interactions and the way structural information is introduced. For instance, TokenGT Kim et al. (2022) treats nodes and edges as input tokens and performs global attention, Graphormer Ying et al. (2021) injects shortest-path and degree information into node representations and attention biases, GraphGPS Rampášek et al. (2022) combines local message passing with global attention and positional encodings, while ESA Buterez et al. (2025) alternates between adjacency-masked and global attention. However, these models often incur a heavy computational burden during training.

2.2 SNNs and their Extension to Graphs

SNNs Maass (1997) communicate through discrete spikes and have been investigated extensively in computer vision Yao et al. (2023a; 2024); Lei et al. (2025); Pan et al. (2025). SpikingGCN Zhu et al. (2022) combines graph convolution with spiking computation, while SpikeGCL Li et al. (2023) learns binary graph representations through contrastive learning. MSG Sun et al. (2024a) further explores spiking graph learning on Riemannian manifolds. Recent studies also introduce spiking attention into graph Transformers Sun et al. (2024b); Qian & Tao (2025). In particular, SpikeGraphormer combines a spiking attention branch with a separate sparse GNN branch, incorporating local graph structure through the fusion of their outputs. Different from these studies, our work focuses on constructing spiking counterparts of existing real-valued MPNNs and graph Transformers, with the goal of preserving the effective architectural designs of their continuous-valued counterparts while enabling spike-based sparse computation.

2.3 Neuromorphic Hardware and Energy Estimation

Recent neuromorphic platforms support hybrid computation, where conventional MAC operations can coexist with spike-triggered accumulation (AC) operations Höppner et al. (2021); Di Mauro et al. (2022). Accordingly, a common approach for estimating the energy consumption of SNNs is to separately count MAC and AC operations and weight them by their corresponding unit energy costs Rathi & Roy (2023); Yao et al. (2023b); Zhou et al. (2022); Yao et al. (2023a; 2024); Lei et al. (2025). Following this widely adopted approach, we estimate the energy consumption of our spiking and non-spiking models based on their MAC and AC operation counts.

3 Methodology

Formally, let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, X)$ denote a graph, where $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ represents the set of N nodes, and $\mathcal{E}$ collects all of the edges. The node feature matrix $X = [x_1, x_2, \cdots, x_N] \in \mathbb{R}^{N \times d}$ records the feature vector $x_i \in \mathbb{R}^d$ for each node v_i. Next, we elaborate on the frameworks for spike-based MPNNs and graph Transformers.

3.1 Spike-based MPNNs

The linear transformation module constitutes the major computational bottleneck in MPNNs Wu et al. (2019). To mitigate this, we redesign the MPNN architecture with the following adjustments:

- We place spiking neurons at the end of each layer, ensuring that the signals transmitted to the subsequent layer are binary spike sequences.
- We reposition the linear layer to the beginning of the MPNN block.

Figure 2 illustrates the architecture of our proposed method. This arrangement allows the linear layer to process sparse spike inputs directly and transforms dense MACs into sparse accumulations. To stabilize feature distribution and facilitate neuronal firing, we incorporate a layer normalization module Ba et al. (2016) before spiking neurons. Specifically, for architectures such as GIN Xu et al. (2018) that utilize an MLP for feature aggregation, we additionally inject spiking neurons within the

MLP block. This design ensures that the inputs to every linear layer throughout the model are spike sequences. Comprehensive architectural details are provided in Appendix A.

Formally, let $\mathbf{S}_t^{(l-1)} \in \{0,1\}^{N\times d}$ be the spike input received by the l-th layer at time t, the aforementioned architecture can be formalize as

$$\mathbf{H}_t^l = \mathrm{AGG}\left(\mathrm{Linear}\left(\mathbf{S}_t^{(l-1)}\right), \mathcal{E}\right), \tag{1}$$

$$\mathbf{I}_t^l = \mathcal{LN}\left(\mathbf{H}_t^l\right), \quad \mathbf{S}_t^l = \mathcal{SN}\left(\mathbf{I}_t^l\right), \tag{2}$$

where AGG denotes the message aggregation mechanism. Different MPNN variants employ distinct aggregation strategies. $\mathcal{LN}$ denotes the layer normalization. $\mathbf{I}_t^l = [\mathbf{i}_{t1}^l, \mathbf{i}_{t2}^l, \cdots, \mathbf{i}_{tN}^l]^\top \in \mathbb{R}^{N\times d}$ denotes the input of spiking neurons. We uniformly implement the spiking neuron $\mathcal{SN}$ using the leaky integrate-and-fire (LIF) model Gerstner & Kistler (2002). Specifically, the output spike $\mathbf{S}_t^l$ of the l-th layer at time step t is jointly determined by the decayed residual potential from the previous time step $\tau\mathbf{U}_{t-1}^l$ and the current input $\mathbf{I}_t^l$, where $\tau \in (0,1)$ serves as the decay factor. When the resulting transient membrane potential $\tilde{\mathbf{U}}_t^l$ (the sum of $\tau\mathbf{U}_{t-1}^l$ and $\mathbf{I}_t^l$) exceeds the firing threshold U_{th}, the spiking neurons generate spikes and the membrane potential $\mathbf{U}_t^l$ is reset, *i.e.*,

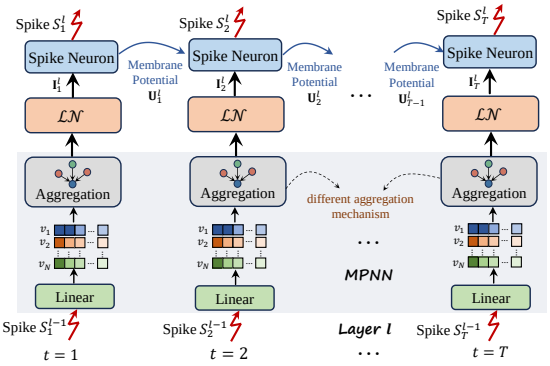

Figure 2: Integration of SNNs with MPNNs. Spike outputs depend on previous membrane potentials and current inputs.

$$\tilde{\mathbf{U}}_t^l = \tau\mathbf{U}_{t-1}^l + \mathbf{I}_t^l, \tag{3}$$

$$\mathbf{S}_t^l = \Theta(\tilde{\mathbf{U}}_t^l - U_{\text{th}}), \tag{4}$$

$$\mathbf{U}_t^l = \tilde{\mathbf{U}}_t^l \cdot (1 - \mathbf{S}_t^l), \tag{5}$$

where $\Theta(\cdot)$ is the Heaviside step function.

Theory on mitigating over-smoothing. Recent research has widely adopted Dirichlet energy to quantify the severity of over-smoothing Li et al. (2018); Cai & Wang (2020); Chamberlain et al. (2021); Rusch et al. (2022). Specifically, for any nodes v_i and v_j, after the spiking neurons receive inputs, let $\tilde{\mathbf{u}}_{ti}^l$ and $\tilde{\mathbf{u}}_{tj}^l$ denote the transient membrane potential at time step t and layer l. The local Dirichlet energy that measures the similarity between v_i and v_j is calculated by

$$E_t^l(v_i, v_j) = \frac{1}{d}\|\Theta(\tilde{\mathbf{u}}_{ti}^l - U_{th}) - \Theta(\tilde{\mathbf{u}}_{tj}^l - U_{th})\|_2^2, \tag{6}$$

In this case, a lower value of $E_t^l(v_i, v_j)$ indicates higher similarity between these two nodes. The following Theorem 1 illustrates that the method proposed above can effectively alleviate the over-smoothing problem of MPNNs.

Theorem 1. *Assume that the dimensions of the transient membrane potentials are independent and identically distributed (i.i.d.), following a probability density function $p(\cdot)$ that is continuously differentiable in the neighborhood of the firing threshold U_{th} and satisfies $p(U_{th}) \geq c_{min} > 0$. For any two nodes v_i and v_j, the transient membrane potentials of their corresponding spiking neurons are $\tilde{\mathbf{u}}_{ti}^l, \tilde{\mathbf{u}}_{tj}^l \in \mathbb{R}^d$ at layer l and time step t, the expected energy response of v_i and v_j decays linearly as $\|\boldsymbol{\delta}\|_2 = \|\tilde{\mathbf{u}}_{ti}^l - \tilde{\mathbf{u}}_{tj}^l\|_2 \to 0$, i.e.,*

$$\mathbb{E}\left(E_t^l(v_i, v_j)\right) \propto \mathcal{O}\left(\|\boldsymbol{\delta}\|_2\right), \tag{7}$$

where $\mathbb{E}$ denotes the mathematical expectation.

The detailed proof is provided in Appendix C. This theorem establishes that the SNN architecture maintains a linear response to subtle feature perturbations. In stark contrast, for non-spiking MPNN counterparts where the spiking neuron is replaced by other activation functions (such as ReLU Nair & Hinton (2010)), the model typically exhibits a quadratic energy decay under small

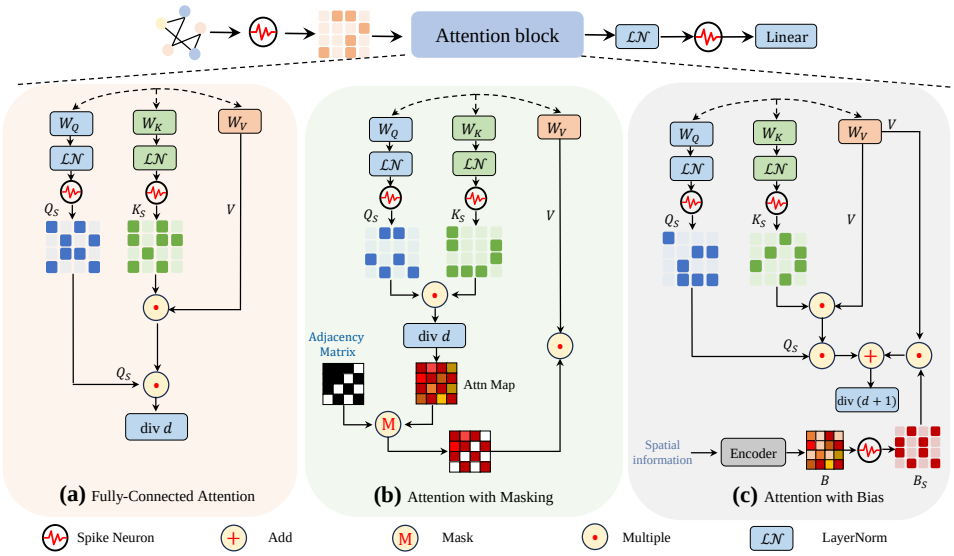

Figure 3: Spiking versions of three attention mechanisms: fully-connected attention (left), adjacency-masked attention (middle), and attention with bias (right).

differences, *i.e.*, $E \propto \mathcal{O}(\|\boldsymbol{\delta}\|_2^2)$ (see Appendix C.2). This quadratic decay implies that subtle variations vanish rapidly, which accelerates the over-smoothing process. Conversely, the linear response of SNNs enables the preservation of node distinctiveness. Mechanistically, the spiking neuron amplifies the minute gaps between membrane potentials near the threshold into distinctly discrete spike signals (0 or 1), effectively preserving the subtle differences between nodes.

3.2 Spike-based Graph Transformer

Our proposed graph Transformer framework also operates over T time steps. For notational brevity, we omit the time subscript t in the following descriptions. Existing graph Transformers incorporate diverse positional encodings. We uniformly denote them as $\mathbf{PE} \in \mathbb{R}^{N \times d}$. The input to the Transformer module is obtained by adding the embedded raw node features X to $\mathbf{PE}$. We convert the input into a spiking sequence Z_S by

$$Z_S = \mathcal{SN}(Z),\ Z = \text{Embed}\,(X) + \mathbf{PE}. \tag{8}$$

At this point, Q, K and V are obtained by computing Z_S with parameters W_K, W_Q, and W_V by

$$Q = W_Q Z_S, \quad K = W_K Z_S, \quad V = W_V Z_S. \tag{9}$$

Subsequently, we convert Q and K into spiking sequences to participate in the calculation of attention scores by

$$Q_S = \mathcal{SN}(\mathcal{LN}(Q)), \quad K_S = \mathcal{SN}(\mathcal{LN}(K)). \tag{10}$$

Note that we only convert Q_S and K_S into spike sequences, while keeping V as a floating-point vector. This strategy ensures the rationality of the model while minimizing information loss during training. The spike-based self-attention mechanism is computed by the following equation:

$$\mathbf{A} = \mathcal{F}(Q_S, K_S, V), \tag{11}$$

where $\mathcal{F}$ denotes the attention computation function. Next, we will introduce the generalized spike-based attention mechanism (fully-connected attention). Building upon this foundation, we further extend our framework to support attention with adjacency matrix masking and attention with bias. The architectures are illustrated in Figure 3.

Fully-Connected Attention. Fully-connected attention is the most fundamental form of the self-attention module, which requires computing the attention between all pairs of nodes Kim et al.

(2022). Let $q_i, k_j \in \{0,1\}^d$ denote the query and key vectors for nodes v_i and v_j. The attention score is computed via bitwise operations, *i.e.*,

$$(Q_S K_S^T)_{ij} = \sum_{m=1}^{d} q_{im} \wedge k_{jm}, \tag{12}$$

where $\wedge$ represents the logical AND operation. In such a way, a higher value of $(Q_S K_S^T)_{ij}$ indicates that q_i and k_j have spike emissions simultaneously in more dimensions, reflecting a strong synchronization and correlation between nodes v_i and v_j. Due to the sparsity of spikes, the resulting attention scores tend to be small and confined to a narrow range. Consequently, applying the standard softmax function yields a near-uniform distribution, thereby stripping the attention mechanism of its selectivity. Therefore, we discard the softmax function. Instead, we pass the aggregated results through a layer normalization module to stabilize the feature distribution, *i.e.*,

$$\mathbf{A} = \mathcal{LN}\left(\frac{Q_S K_S^T}{d} V\right), \tag{13}$$

where d serves as a normalization factor to bound the attention scores within the range of $[0, 1]$. By discarding the softmax, we take advantage of the associativity of matrix multiplication to compute $Q_S(K_S^T V)$ instead of $(Q_S K_S^T)V$ (see Figure 3 (a)). This reordering achieves linear complexity $\mathcal{O}(N)$ with respect to the number of nodes N, in contrast to the $\mathcal{O}(N^2)$ complexity of the vanilla self-attention. Furthermore, since Q_S and K_S are binary spikes, these operations are transformed into energy-efficient sparse accumulations.

Attention with Adjacency Matrix Masking. For attention with adjacency matrix masking, we incorporate the adjacency matrix $\mathbf{M}$ into the computation to filter out non-neighboring connections:

$$\mathbf{A} = \mathcal{LN}\left(\frac{Q_S K_S^T}{d} \odot \mathbf{M} \cdot V\right), \tag{14}$$

where $\odot$ denotes the Hadamard product. If there is an edge between node v_i and node v_j, then $\mathbf{M}_{ij} = 1$; otherwise, $\mathbf{M}_{ij} = 0$. At this point, the attention between non-directly connected nodes is directly set to 0 (see Figure 3(b)). Although the Hadamard product with $\mathbf{M}$ restricts the reordering of matrix operations (normally implying $\mathcal{O}(N^2)$ complexity), for sparse graphs where $|\mathcal{E}| \ll N^2$, the complexity can be reduced to $\mathcal{O}(N)$.

Attention with Bias. Attention bias (B) refers to an $N \times N$ matrix that explicitly encodes spatial information between nodes. The bias is directly added to the original attention scores to yield the new attention scores. We can similarly convert the bias matrix into the spiking sequence B_S by

$$B_S = \mathcal{SN}(B), \quad B_{ij} = \text{SpatialEncoder}(v_i, v_j), \tag{15}$$

$$\mathbf{A} = \mathcal{LN}\left(\frac{Q_S K_S^T + B_S}{d+1} \cdot V\right), \tag{16}$$

where the denominator is set to $d + 1$ to keep the attention scores fall within $[0, 1]$. During computation, we apply V to $Q_S K_S^T$ and B_S separately (see Figure 3 (c)), ensuring that the entire process is executed via sparse addition operations.

4 Experiments

Datasets. For a broader evaluation, we employ both graph regression and classification tasks. For regression, we select ESOL Delaney (2004), FreeSolv Mobley & Guthrie (2014), and Lipo Gaulton et al. (2012); for classification, we choose NCI1 Wale et al. (2008), ENZYMES Borgwardt et al. (2005), and BACE Subramanian et al. (2016). Detailed information is provided in Appendix B.1.

Models. To make the comparison more convincing, our baseline selection covers a diverse range of architectures. For MPNNs, we select six canonical models representing different aggregation strategies, including GCN Kipf & Welling (2016), GraphConv Morris et al. (2019), GraphSAGE Hamilton et al. (2017), GIN Xu et al. (2018), GAT Velickovic et al. (2017), and GATv2 Brody et al. (2021). For graph Transformers, we specifically choose representative models corresponding to the three self-attention mechanisms: TokenGT Kim et al. (2022) for fully-connected attention, ESA Buterez et al. (2025) for axttention with masking, and Graphormer Ying et al. (2021)

Table 1: Performance comparison on six datasets. **Bold** indicates the best performance across all models for each dataset. Background colors indicate the winner in pairwise comparisons (Spiking vs. Non-Spiking): Blue denotes the spiking version performs better, while Red denotes the non-spiking version performs better. **RMSE** (Root Mean Square Error): Lower is better. **ACC** (Accuracy): Higher is better. **OOM**: Out of Memory. All results are reported as the mean ± standard deviation over 5 independent runs.

	Model	Spiking	ESOL (RMSE ↓)	Lipo (RMSE ↓)	FreeSolv (RMSE ↓)	NCI1 (Acc% ↑)	ENZYMES (Acc% ↑)	BACE (Acc% ↑)
Baseline	SpikingGCN	✓	-	-	-	71.86 ± 0.28	48.74 ± 6.85	74.38 ± 2.56
	SpikeGCL	✓	0.695 ± 0.049	0.716 ± 0.019	1.004 ± 0.161	81.31 ± 0.37	41.86 ± 12.56	68.09 ± 7.31
	SpikeGraphormer	✓	0.908 ± 0.054	1.044 ± 0.064	1.616 ± 0.080	72.57 ± 2.10	63.45 ± 4.45	68.91 ± 1.36
MPNN	GCN	✗	0.801 ± 0.039	0.760 ± 0.015	1.257 ± 0.051	77.58 ± 1.35	47.51 ± 2.56	64.47 ± 3.89
		✓	0.773 ± 0.034	0.739 ± 0.013	1.157 ± 0.175	83.26 ± 1.08	49.18 ± 4.53	72.70 ± 3.74
	GraphConv	✗	0.678 ± 0.034	0.661 ± 0.021	1.060 ± 0.144	81.59 ± 0.68	52.34 ± 4.20	72.92 ± 1.28
		✓	0.695 ± 0.070	0.644 ± 0.013	1.093 ± 0.155	**84.47** ± 0.94	56.06 ± 4.47	75.66 ± 1.20
	GraphSAGE	✗	0.685 ± 0.080	0.654 ± 0.017	1.062 ± 0.063	83.54 ± 0.82	58.47 ± 3.50	72.80 ± 1.91
		✓	0.669 ± 0.046	0.647 ± 0.023	0.969 ± 0.170	83.66 ± 1.12	55.94 ± 3.61	73.68 ± 0.88
	GIN	✗	0.612 ± 0.028	0.709 ± 0.028	1.182 ± 0.252	77.69 ± 1.51	42.90 ± 4.35	69.87 ± 6.57
		✓	0.584 ± 0.033	0.686 ± 0.027	1.018 ± 0.142	80.05 ± 1.38	50.00 ± 7.39	73.03 ± 2.01
	GAT	✗	0.668 ± 0.015	0.676 ± 0.020	1.085 ± 0.069	80.54 ± 1.81	56.86 ± 6.15	73.90 ± 1.81
		✓	0.665 ± 0.033	0.606 ± 0.016	0.841 ± 0.106	82.60 ± 0.73	63.52 ± 5.98	76.10 ± 1.60
	GATv2	✗	0.657 ± 0.045	0.639 ± 0.011	1.043 ± 0.056	82.20 ± 1.32	55.13 ± 4.27	77.30 ± 2.14
		✓	0.651 ± 0.037	**0.606** ± 0.006	0.933 ± 0.082	83.01 ± 0.68	62.95 ± 1.81	76.18 ± 2.10
Graph Transformer	TokenGT	✗	0.622 ± 0.024	0.883 ± 0.032	1.215 ± 0.258	77.49 ± 1.53	63.97 ± 2.82	76.81 ± 0.99
		✓	0.613 ± 0.020	0.838 ± 0.036	1.061 ± 0.110	77.53 ± 0.89	66.33 ± 3.69	75.13 ± 2.48
	Graphormer	✗	0.606 ± 0.033	OOM	1.051 ± 0.104	76.00 ± 1.42	62.67 ± 5.70	74.47 ± 2.10
		✓	0.648 ± 0.022	OOM	0.820 ± 0.081	81.59 ± 0.92	64.60 ± 4.82	75.43 ± 2.87
	ESA	✗	0.546 ± 0.042	0.671 ± 0.017	0.907 ± 0.160	83.09 ± 1.26	67.46 ± 1.65	**77.63** ± 1.32
		✓	**0.523** ± 0.037	0.670 ± 0.021	**0.776** ± 0.062	82.67 ± 0.93	**68.14** ± 1.76	76.32 ± 1.61

for attention with bias. To demonstrate the effectiveness of our proposed method, we design both spiking and non-spiking versions for each model. Furthermore, we also select existing spike-based GNNs for comparison, including SpikingGCN Zhu et al. (2022), SpikeGCL Li et al. (2023), and SpikeGraphormer Sun et al. (2024b). Detailed experimental settings are provided in Appendix B.2.

4.1 Performance

Table 1 presents the comprehensive performance comparison between spiking and non-spiking versions across six datasets. Experimental results demonstrate that the spiking versions outperform their non-spiking counterparts in **45** out of **53** reported experimental settings (approx. **85%**). These gains validate that, unlike the single-time-step inference of non-spiking versions, spike-based models can leverage multiple time steps to perform graph learning, endowing the networks with stronger feature extraction capability and noise robustness. However, it cannot be ignored that the process of converting continuous values into discrete spikes inevitably introduces information loss, especially when T is small. This explains specific cases where spiking models underperform, such as GraphConv on ESOL (higher RMSE by **0.017**). Furthermore, compared to existing spiking graph neural networks, our model demonstrates advantages. For instance, on the ESOL dataset, our spiking Graphormer achieves an RMSE of **0.648**, significantly outperforming SpikeGraphormer (**0.908**).

4.2 Energy Computation

For spike-based models, the spike firing rate affects the number of computations performed by the models. We calculate the spike firing rates during the models' forward propagation. The results are presented in Appendix D. The firing rate of the vast majority of neurons remains below **10%**. This indicates that the spike sequence has extremely high sparsity and also reflects the feasibility of energy saving during the training process. To quantify this, we follow the theoretical energy calculation method proposed in Yao et al. (2023a). We count the energy consumption required for the forward propagation of a batch of data from the ESOL dataset. Figure 4 presents the results. The results indicate a significant reduction in energy consumption upon the introduction of spiking neurons, with the spiking versions requiring only **6% to 19%** of the energy consumed by their non-spiking counterparts. Notably, in the non-spiking versions of MPNN models, linear layers account

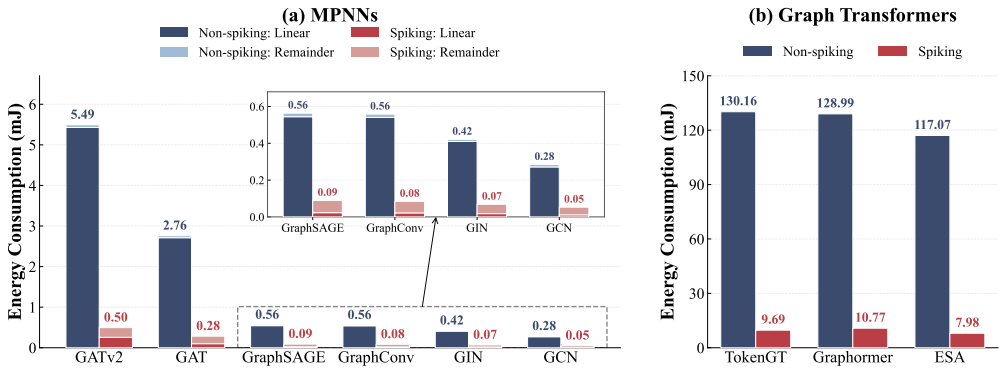

Figure 4: Estimated energy consumption of the MPNN and graph Transformer models. (a) For MPNNs, linear layers dominate the energy consumption of the non-spiking models, while this cost is substantially reduced in the spiking versions. (b) For graph Transformers, the spiking versions also exhibit markedly lower estimated energy consumption than their non-spiking counterparts.

Table 2: Ablation study results using the softmax instead of LayerNorm. The red arrows indicate the performance gap (degradation) compared to the LayerNorm implementation. The results demonstrate that softmax performs worse than LayerNorm.

Model	ESOL (RMSE ↓)	Lipo (RMSE ↓)	FreeSolv (RMSE ↓)	NCI1 (Acc% ↑)	ENZYMES (Acc% ↑)	BACE (Acc% ↑)
TokenGT	0.857 ↑0.244	0.916 ↑0.078	1.441 ↑0.380	75.40 ↓2.13	64.51 ↓1.82	61.75 ↓13.38
Graphormer	0.732 ↑0.084	OOM	1.385 ↑0.565	73.19 ↓8.40	63.40 ↓1.20	72.89 ↓5.40
ESA	0.556 ↑0.033	0.764 ↑0.094	0.826 ↑0.050	81.90 ↓0.74	66.11 ↓2.03	75.79 ↓0.53

for the vast majority of energy consumption (more than **90%**). Upon introducing our method, this portion of energy consumption is drastically reduced.

4.3 Ablation Study

When designing the graph Transformer model, we replace the conventional softmax function with a layer normalization module. To validate the effectiveness of this design, we conduct an ablation study comparing our method against a baseline that retains the softmax function. The results, presented in Table 2, show that our proposed method achieves superior performance. In contrast, reverting to softmax leads to performance degradation, demonstrating that layer normalization is more effective than the softmax at stabilizing feature distributions within the spiking attention mechanism.

4.4 Over-Smoothing Property

To investigate the effect of SNNs on alleviating the over-smoothing problem of MPNNs, we visualize the evolution of node similarity across different numbers of layers. The results of the GCN model are presented in Figure 5. It can be observed that for non-spiking models, the node features become completely identical when the number of layers reaches 4. In contrast, the spike-based models can still distinguish between nodes even when the number of layers increases to 16. Analyzing from the perspective of Dirichlet energy, the non-spiking GCN exhibits a sharp decline, dropping from **1101.23** to **2.32** by just the 4-th layer (a 99.79% drop). In contrast, the spike-based GCN maintains significant energy even at the 16-th layer, decreasing only from **258.01** to **137.65** (a 46.65% drop).

4.5 The Impact of T

The output of the model is jointly determined by the results across T time steps. To investigate the impact of T, we conduct experiments using the ESOL dataset, with T ranging from 1 to 100. The results are presented in Figure 6. With a small T, spiking models generally underperform their non-spiking counterparts due to information loss caused by spike encoding. As T increases, the per-

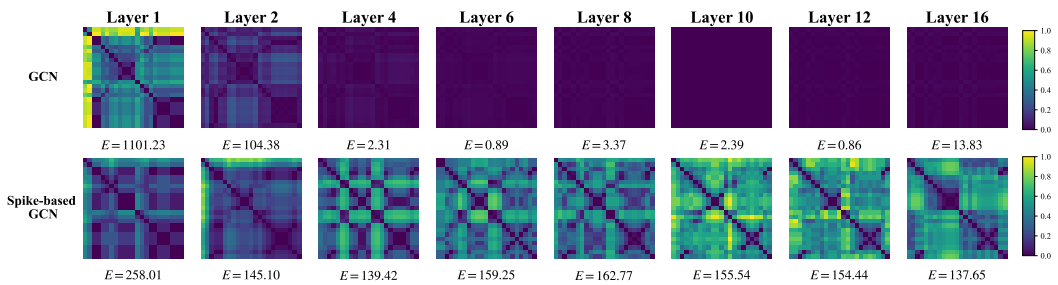

Figure 5: Evolution of node feature distance in standard GCN vs Spike-based GCN as the number of layers increases. The heatmaps display the normalized Euclidean distance matrices of node features. E denotes Dirichlet Energy. For standard GCN, the node features become identical when the number of layers reaches 4. In contrast, spike-based GCN consistently maintains distinguishability.

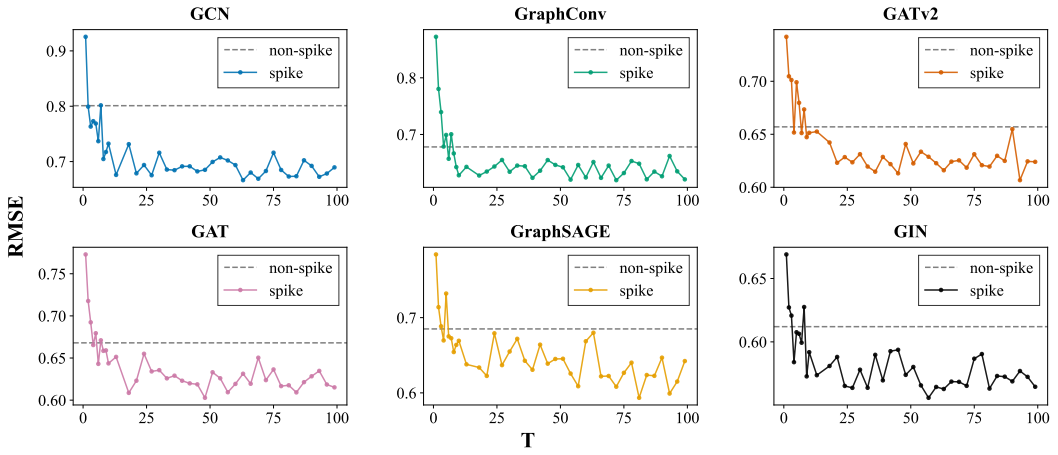

Figure 6: Performance variation of the MPNN models on the ESOL dataset with the increase of T. The model's performance does not improve indefinitely as T increases.

formance improves, eventually surpassing the non-spiking counterparts and reaching convergence. However, since computational cost scales linearly with T, an excessively large T will diminish the energy-efficiency advantages of SNNs. For this reason, we select $T = 4$ in our experiments.

5 Conclusion

We propose a framework for integrating SNNs into various GNNs. For MPNN models, our method can convert the most energy-intensive component into sparse spike-based addition operations and effectively alleviate the over-smoothing problem of MPNN models. For graph Transformer models, we design a spike-based self-attention mechanism that can be effectively extended to various graph Transformer architectures. Experimental results demonstrate that our method significantly reduces energy consumption without sacrificing model performance.

Limitation and Future Work. A limitation of the current framework is that temporal unrolling over multiple time steps introduces additional memory and computational overhead, which may affect scalability on very large graphs. This limitation also highlights the importance of hardware-aware implementation for fully exploiting the sparse and event-driven nature of SNNs. In future work, we plan to extend our framework to neuromorphic hardware and systematically evaluate its practical efficiency in such environments.

AI USE STATEMENT

Generative AI tools were used only for language polishing and improving the readability of the manuscript. All AI-assisted edits were reviewed by the authors, who take full responsibility for the final content.

REFERENCES

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1): i47–i56, 2005.

Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*, 2021.

David Buterez, Jon Paul Janet, Dino Oglic, and Pietro Liò. An end-to-end attention-based approach for learning on graphs. *Nature Communications*, 16(1):5244, 2025.

Chen Cai and Yusu Wang. A note on over-smoothing for graph neural networks. *arXiv preprint arXiv:2006.13318*, 2020.

Ben Chamberlain, James Rowbottom, Maria I Gorinova, Michael Bronstein, Stefan Webb, and Emanuele Rossi. Grand: Graph neural diffusion. In *International Conference on Machine Learning*, pp. 1407–1418. PMLR, 2021.

Dexiong Chen, Leslie O'Bray, and Karsten Borgwardt. Structure-aware transformer for graph representation learning. In *International conference on machine learning*, pp. 3469–3489. PMLR, 2022.

John S Delaney. Esol: estimating aqueous solubility directly from molecular structure. *Journal of Chemical Information and Computer Sciences*, 44(3):1000–1005, 2004.

Alfio Di Mauro, Moritz Scherer, Davide Rossi, and Luca Benini. Kraken: A direct event/frame-based multi-sensor fusion soc for ultra-efficient visual processing in nano-uavs. *arXiv preprint arXiv:2209.01065*, 2022.

Yongqi Ding, Lin Zuo, Mengmeng Jing, Kunshan Yang, Pei He, and Tonglan Xie. Synergy between the strong and the weak: Spiking neural networks are inherently self-distillers. *arXiv preprint arXiv:2510.07924*, 2025.

Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.

Anna Gaulton, Louisa J Bellis, A Patricia Bento, Jon Chambers, Mark Davies, Anne Hersey, Yvonne Light, Shaun McGlinchey, David Michalovich, Bissan Al-Lazikani, et al. Chembl: a large-scale bioactivity database for drug discovery. *Nucleic Acids Research*, 40(D1):D1100–D1107, 2012.

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

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pp. 1263–1272. Pmlr, 2017.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, 30, 2017.

Sebastian Höppner, Yexin Yan, Andreas Dixius, Stefan Scholze, Johannes Partzsch, Marco Stolba, Florian Kelber, Bernhard Vogginger, Felix Neumärker, Georg Ellguth, Stephan Hartmann, Stefan Schiefer, Thomas Hocker, Dennis Walter, Gengting Liu, Jim Garside, Steve Furber, and Christian Mayr. The spinnaker 2 processing element architecture for hybrid digital neuromorphic computing. *arXiv preprint arXiv:2103.08392*, 2021.

Jinwoo Kim, Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. Pure transformers are powerful graph learners. *Advances in Neural Information Processing Systems*, 35:14582–14595, 2022.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Zhenxin Lei, Man Yao, Jiakui Hu, Xinhao Luo, Yanye Lu, Bo Xu, and Guoqi Li. Spike2former: Efficient spiking transformer for high-performance image segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 1364–1372, 2025.

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. *arXiv preprint arXiv:2305.19306*, 2023.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Yanyu Li, Geng Yuan, Yang Wen, Ju Hu, Georgios Evangelidis, Sergey Tulyakov, Yanzhi Wang, and Jian Ren. Efficientformer: Vision transformers at mobilenet speed. *Advances in Neural Information Processing Systems*, 35:12934–12949, 2022.

Ziqi Li, Tao Gao, Yisheng An, Ting Chen, Jing Zhang, Yuanbo Wen, Mengkun Liu, and Qianxi Zhang. Brain-inspired spiking neural networks for energy-efficient object detection. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 3552–3562, 2025.

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

David L Mobley and J Peter Guthrie. Freesolv: a database of experimental and calculated hydration free energies, with input files. *Journal of Computer-Aided Molecular Design*, 28(7):711–720, 2014.

Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4602–4609, 2019.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 807–814, 2010.

Yi Pan, Hanqi Jiang, Junhao Chen, Yiwei Li, Huaqin Zhao, Yifan Zhou, Peng Shu, Zihao Wu, Zhengliang Liu, Dajiang Zhu, et al. Eg-spikeformer: Eye-gaze guided transformer on spiking neural networks for medical image analysis. In *2025 IEEE 22nd International Symposium on Biomedical Imaging (ISBI)*, pp. 1–5. IEEE, 2025.

Jun Qian and Haotian Tao. Sgt: an energy-efficient spiking graph transformer for graph representation learning. In *Fifth International Conference on Image Processing and Intelligent Control (IPIC 2025)*, volume 13782, pp. 360–366. SPIE, 2025.

Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35:14501–14515, 2022.

Yongming Rao, Zuyan Liu, Wenliang Zhao, Jie Zhou, and Jiwen Lu. Dynamic spatial sparsification for efficient vision transformers and convolutional neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(9):10883–10897, 2023.

Nitin Rathi and Kaushik Roy. Diet-snn: A low-latency spiking neural network with direct input encoding and leakage and threshold optimization. *IEEE Transactions on Neural Networks and Learning Systems*, 34(6):3174–3182, 2023. doi: 10.1109/TNNLS.2021.3111897.

T Konstantin Rusch, Ben Chamberlain, James Rowbottom, Siddhartha Mishra, and Michael Bronstein. Graph-coupled oscillator networks. In *International Conference on Machine Learning*, pp. 18888–18909. PMLR, 2022.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

Govindan Subramanian, Bharath Ramsundar, Vijay Pande, and Rajiah Aldrin Denny. Computational modeling of β-secretase 1 (bace-1) inhibitors using ligand based approaches. *Journal of Chemical Information and Modeling*, 56(10):1936–1949, 2016.

Li Sun, Zhenhao Huang, Qiqi Wan, Hao Peng, and Philip S Yu. Spiking graph neural network on riemannian manifolds. *Advances in Neural Information Processing Systems*, 37:34025–34055, 2024a.

Yundong Sun, Dongjie Zhu, Yansong Wang, Zhaoshuo Tian, Ning Cao, and Gregory O'Hared. Spikegraphormer: A high-performance graph transformer with spiking graph attention. *arXiv preprint arXiv:2403.15480*, 2024b.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. Graph attention networks. *Stat*, 1050(20):10–48550, 2017.

Nikil Wale, Ian A Watson, and George Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems*, 14(3):347–375, 2008.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International Conference on Machine Learning*, pp. 6861–6871. Pmlr, 2019.

Yujie Wu, Rong Zhao, Jun Zhu, Feng Chen, Mingkun Xu, Guoqi Li, Sen Song, Lei Deng, Guanrui Wang, Hao Zheng, et al. Brain-inspired global-local learning incorporated with neuromorphic computing. *Nature Communications*, 13(1):65, 2022.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Mingkun Xu, Yujie Wu, Lei Deng, Faqiang Liu, Guoqi Li, and Jing Pei. Exploiting spiking dynamics with spatial-temporal feature normalization in graph learning. *arXiv preprint arXiv:2107.06865*, 2021.

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

Man Yao, Guangshe Zhao, Hengyu Zhang, Yifan Hu, Lei Deng, Yonghong Tian, Bo Xu, and Guoqi Li. Attention spiking neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(8):9393–9410, 2023b. doi: 10.1109/TPAMI.2023.3241201.

Man Yao, Jiakui Hu, Tianxiang Hu, Yifan Xu, Zhaokun Zhou, Yonghong Tian, Bo Xu, and Guoqi Li. Spike-driven transformer v2: Meta spiking neural network architecture inspiring the design of next-generation neuromorphic chips. *arXiv preprint arXiv:2404.03663*, 2024.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems*, 34:28877–28888, 2021.

Qianmu Yuan, Sheng Chen, Jiahua Rao, Shuangjia Zheng, Huiying Zhao, and Yuedong Yang. Alphafold2-aware protein–dna binding site prediction using graph transformer. *Briefings in Bioinformatics*, 23(2):bbab564, 2022.

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

Zulun Zhu, Jiaying Peng, Jintang Li, Liang Chen, Qi Yu, and Siqiang Luo. Spiking graph convolutional networks. *arXiv preprint arXiv:2205.02767*, 2022.

Jiaming Zhuo, Yuwei Liu, Yintong Lu, Ziyi Ma, Kun Fu, Chuan Wang, Yuanfang Guo, Zhen Wang, Xiaochun Cao, and Liang Yang. Dualformer: Dual graph transformer. In *The Thirteenth International Conference on Learning Representations*, 2025.

A MLP

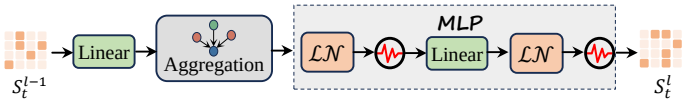

Figure 7: Architecture of MPNN models requiring MLP modules. Note that the linear layers within the MLP also receive spike sequences as inputs.

For models such as GIN that rely on MLP modules for aggregation, we have specifically designed a spike-based MLP module. Specifically, we adopt a similar strategy by placing spiking neurons immediately preceding the linear layers. The architecture diagram is illustrated in Figure 7.

B Dataset Information and Experimental Settings

B.1 Datasets

Table 3 presents detailed statistics for the six datasets used in the experiments. The average number of edges connected to each node is significantly smaller than the average number of nodes per graph, indicating that these are all sparse graphs. Therefore, if the adjacency matrix is used to mask the attention mechanism, a computational complexity of $\mathcal{O}(N)$ can be achieved.

Table 3: Relationship between the number of nodes and edges in each dataset. The average number of nodes per graph is the total number of nodes in the dataset divided by the number of graphs, and the average number of edges per node is the total number of edges in the dataset divided by the total number of nodes.

	NCI1	**BACE**	**ENZYMES**	**FreeSolv**	**Lipo**	**ESOL**
Total graphs	4110	1513	600	642	4200	1128
Total nodes	122,747	51,577	19,580	5,600	113,568	14,991
Total edges	265,506	111,536	74,564	10,770	247,798	30,856
Ave. nodes per graph	29.9	34.1	32.6	8.7	27.0	13.3
Ave. edges per node	2.16	2.16	3.81	1.92	2.18	2.06

B.2 Experimental Settings

To ensure fairness, we configure a unified set of hyperparameters. For MPNN models, we set the number of layers to 3. For graph Transformer models, we set the number of layers to 8. The hidden layer dimension (d) is set to 256. The batch size (B) is set to 32. For spike-based models, we set the number of time steps (T) to 4. The firing threshold U_{th} is set to 1, and the decay factor τ is set to 0.5. [1]

C Proof

C.1 Linear Response of Spiking MPNNs

To facilitate the proof, we omit the layer and time indices. Specifically, let $\tilde{\mathbf{u}}_i, \tilde{\mathbf{u}}_j \in \mathbb{R}^d$ denote the transient membrane potentials of spiking neurons for nodes v_i and v_j, respectively. For any dimension k, the output spikes of nodes v_i and v_j differ (*i.e.*, $|s_i^{(k)} - s_j^{(k)}| = 1$) if and only if the threshold U_{th} falls between the membrane potentials $\tilde{\mathbf{u}}_i^{(k)}$ and $\tilde{\mathbf{u}}_j^{(k)}$, *i.e.*,

$$|s_i^{(k)} - s_j^{(k)}| = 1 \iff U_{th} \in [\min(\tilde{\mathbf{u}}_i^{(k)}, \tilde{\mathbf{u}}_j^{(k)}), \max(\tilde{\mathbf{u}}_i^{(k)}, \tilde{\mathbf{u}}_j^{(k)})] \tag{17}$$

Given that the inputs to the spiking neurons follow a probability density function $p(\cdot)$, as $\delta^{(k)}$ approaches 0, we have:

$$\begin{aligned} P\left(|s_i^{(k)} - s_j^{(k)}| = 1\right) &= \int_{\min(\tilde{\mathbf{u}}_i^{(k)}, \tilde{\mathbf{u}}_j^{(k)})}^{\max(\tilde{\mathbf{u}}_i^{(k)}, \tilde{\mathbf{u}}_j^{(k)})} p(x)\, dx \\ &\approx p(U_{th}) \cdot \left|\tilde{\mathbf{u}}_i^{(k)} - \tilde{\mathbf{u}}_j^{(k)}\right| \\ &= p(U_{th}) \cdot \left|\delta^{(k)}\right| \end{aligned} \tag{18}$$

In this case, the corresponding expected value of the Dirichlet energy is given by:

$$\begin{aligned} \mathbb{E}[E(v_i, v_j)] &= \frac{1}{d} \sum_{k=1}^{d} \mathbb{E}\left[(s_i^{(k)} - s_j^{(k)})^2\right] \\ &\approx \frac{1}{d} \sum_{k=1}^{d} p(U_{th}) \cdot |\delta^{(k)}| \\ &= \frac{p(U_{th})}{d} \|\boldsymbol{\delta}\|_1 \end{aligned} \tag{19}$$

Using the norm inequality $\|\boldsymbol{\delta}\|_1 \leq \sqrt{d}\|\boldsymbol{\delta}\|_2$, we have:

$$\mathbb{E}[E(v_i, v_j)] \leq \frac{p(U_{th})}{d} \cdot \sqrt{d}\|\boldsymbol{\delta}\|_2 = \frac{p(U_{th})}{\sqrt{d}} \|\boldsymbol{\delta}\|_2 \tag{20}$$

Therefore, $\mathbb{E}\left(E(v_i, v_j)\right) \propto \mathcal{O}(\|\boldsymbol{\delta}\|_2)$.

C.2 Comparison with Non-Spiking Models

To demonstrate the advantage of the proposed method, we compare it with non-spiking MPNNs using continuous activation functions. Let $\mathbf{i}_i^l, \mathbf{i}_j^l \in \mathbb{R}^d$ donate the input of the activation function $\sigma(\cdot)$ for any pair of nodes v_i, v_j. The the local Dirichlet energy is defined as:

$$E^l(v_i, v_j) = \frac{1}{d}\|\sigma(\mathbf{i}_i^l) - \sigma(\mathbf{i}_j^l)\|_2^2. \tag{21}$$

Then we have Theorem 2:

[1] The code is available at this anonymous repository.

Table 4: Average firing rates of MPNN models across different time steps. The firing rates are consistently below 10%.

Model	Time Steps (T)			
	$T=1$	$T=2$	$T=3$	$T=4$
GCN	2.19%	9.92%	5.41%	9.18%
GraphConv	2.43%	7.97%	5.79%	8.37%
GraphSAGE	2.18%	8.29%	8.04%	7.86%
GIN	2.13%	6.85%	6.31%	6.87%
GAT	1.46%	8.67%	4.24%	7.77%
GATv2	1.79%	8.93%	4.72%	8.95%

Theorem 2. *Assume the activation function $\sigma(\cdot)$ is Lipschitz continuous with a constant L (*e.g.*, for ReLU, $L=1$). For any two nodes v_i and v_j , the inputs to $\sigma(\cdot)$ are $\mathbf{i}_i^l, \mathbf{i}_j^l \in \mathbb{R}^d$, the energy response of v_i and v_j decays quadratically as $\|\boldsymbol{\delta}\|_2 \to 0$, i.e.,*

$$E^l(v_i, v_j) \propto \mathcal{O}\left(\|\boldsymbol{\delta}\|_2^2\right). \tag{22}$$

Proof. To facilitate the proof, we also omit the superscript l. For any dimension k, based on the Lipschitz continuity of $\sigma(\cdot)$, we have:

$$\left|\sigma(\mathbf{i}_i^{(k)}) - \sigma(\mathbf{i}_j^{(k)})\right| \leq L\left|\mathbf{i}_i^{(k)} - \mathbf{i}_j^{(k)}\right| = L\left|\delta^{(k)}\right|. \tag{23}$$

Substituting this into the definition of Dirichlet energy:

$$\begin{aligned} E(v_i, v_j) &= \frac{1}{d}\sum_{k=1}^{d}\left(\sigma(\mathbf{i}_i^{(k)}) - \sigma(\mathbf{i}_j^{(k)})\right)^2 \\ &\leq \frac{1}{d}\sum_{k=1}^{d} L^2\left(\delta^{(k)}\right)^2 \\ &= \frac{L^2}{d}\|\boldsymbol{\delta}\|_2^2. \end{aligned} \tag{24}$$

Therefore, $E(v_i, v_j) \propto \mathcal{O}(\|\boldsymbol{\delta}\|_2^2)$.

This quadratic decay indicates that the energy of non-spiking models vanishes much faster than that of spiking models as nodes become similar, making them more susceptible to over-smoothing.

D Spike Firing Rates

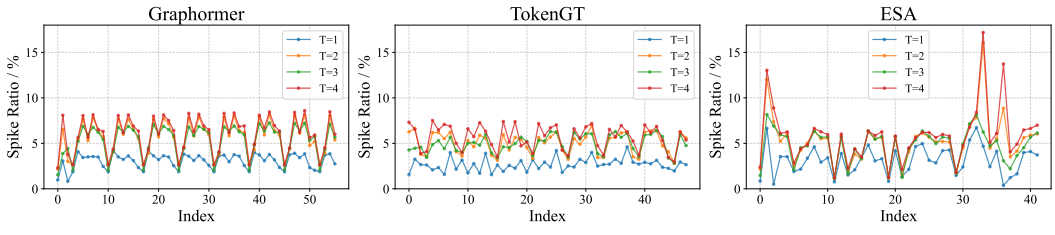

Figure 8: Spike firing rates of the three Graph Transformer models. The x-axis represents the index of the spiking neurons.

Figure 8 and Table 4 illustrate the firing rates across four time steps. It can be observed that only a small fraction of spiking neurons exhibit firing rates exceeding 10%, while the majority remain

below this threshold. This demonstrates the energy-efficient characteristic of spiking neurons. We also observe that the firing rate at $T = 1$ is significantly lower than that at other time steps. This is due to the absence of residual membrane potential from the previous time step at $T = 1$.

E Over-smoothing Characteristics of the Remaining MPNNs

Figure 9 illustrates the evolution of pair-wise node distances for the remaining five MPNN models as the depth of the network increases. Regardless of the architecture, the non-spiking versions succumb to over-smoothing at very shallow depths. In contrast, the spiking counterparts significantly postpone this phenomenon.

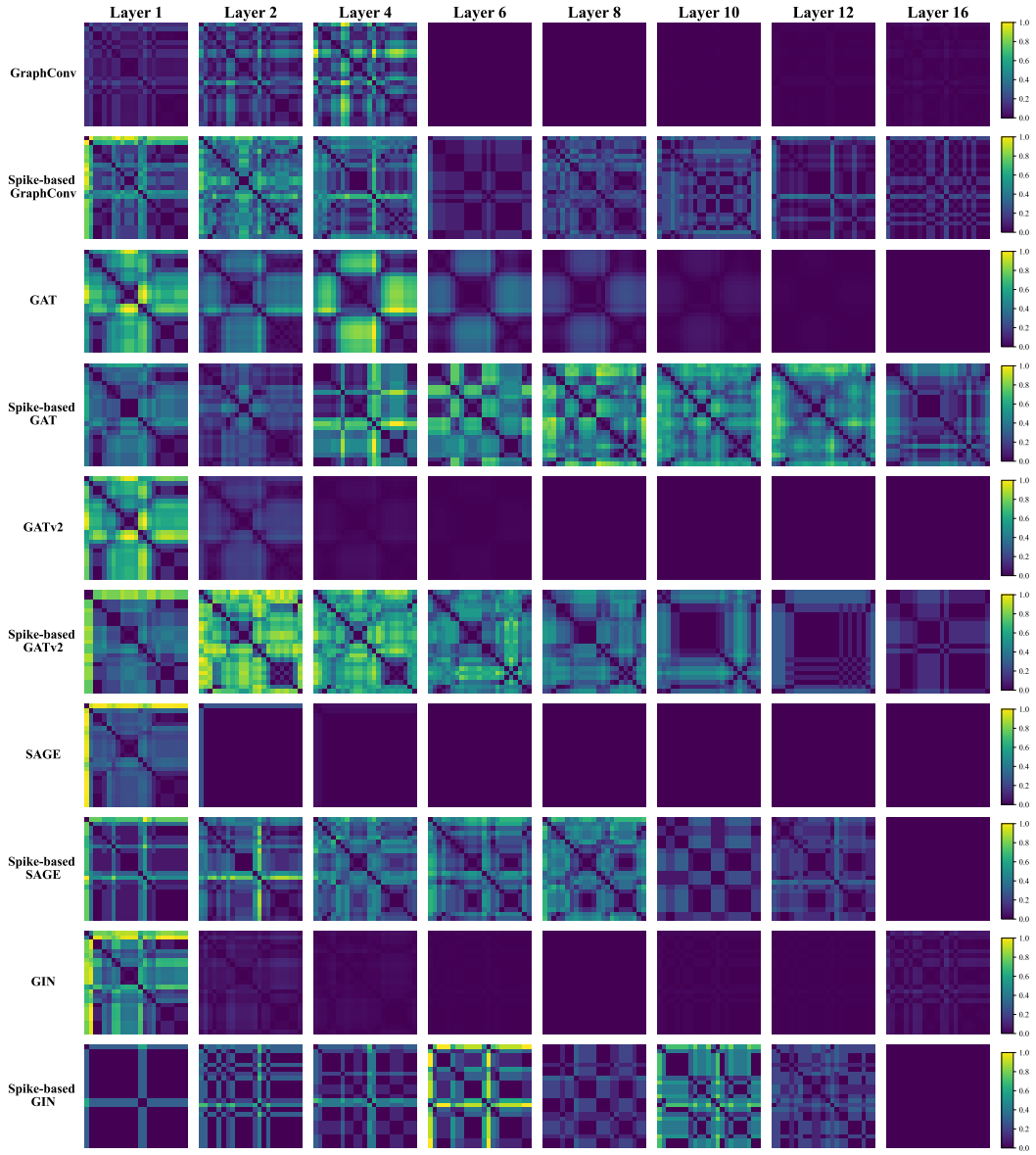

Figure 9: The variations in node similarity between spiking and non-spiking versions of MPNN models as the number of layers increases.