SGHormer: An Energy-Saving Graph Transformer Driven by Spikes

Huizhe Zhang[1] , **Jintang Li**[1] , **Liang Chen**[1] * and **Zibin Zheng**[1]
[1]Sun Yat-sen University
{zhanghzh33, lijt55}@mail2.sysu.edu.cn, {chenliang6, zhzibin}@mail.sysu.edu

Abstract

Graph Transformers (GTs) with powerful representation learning ability make a huge success in wide range of graph tasks. However, the costs behind outstanding performances of GTs are higher energy consumption and computational overhead. The complex structure and quadratic complexity during attention calculation in vanilla transformer seriously hinder its scalability on the large-scale graph data. Though existing methods have made strides in simplifying combinations among blocks or attention-learning paradigm to improve GTs' efficiency, a series of energy-saving solutions originated from biologically plausible structures are rarely taken into consideration when constructing GT framework. To this end, we propose a new spiking-based graph transformer (SGHormer). It turns full-precision embeddings into sparse and binarized spikes to reduce memory and computational costs. The spiking graph self-attention and spiking rectify blocks in SGHormer explicitly capture global structure information and recover the expressive power of spiking embeddings, respectively. In experiments, SGHormer achieves comparable performances to other full-precision GTs with extremely low computational energy consumption. The results show that SGHomer makes a remarkable progress in the field of low-energy GTs. Code is available at https://github.com/Zhhuizhe/SGHormer.

1 Introduction

Graph neural networks (GNNs) as a flourishing representation learning methods on graph data have been developed and applied on diverse tasks [Ying *et al.*, 2018] [Zhu *et al.*, 2023]. Most GNNs based on message passing paradigm can effectively generate representations of nodes by exchange the local structure information among nodes [Hamilton *et al.*, 2018]. Despite message passing neural networks (MPNNs) have strong capabilities in capturing graph inductive biases, there are still some of inherent drawbacks are uncovered and

*Corresponding author.

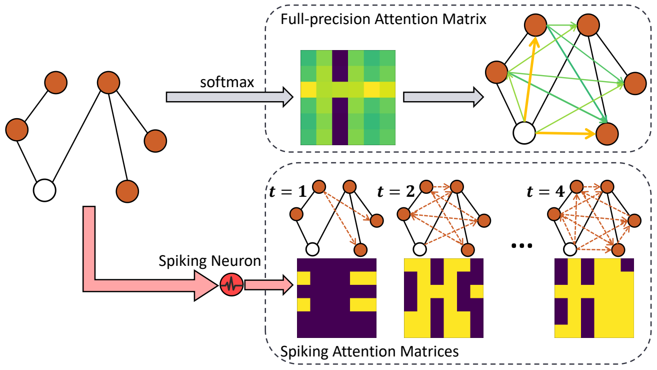

Figure 1: Visualization results of spiking attention matrices and the full-precision vanilla self-attention matrix. We construct the experiments on a selected graph from ZINC. For spiking attention matrices cross multiple time steps (**bottom**), parts of spiking outputs have similar attention patterns as the full-precision attention (**top**) calculated by a softmax function.

formalized such as over-squashing, long-range dependencies and expressive power limitations [Dai *et al.*, 2018] [Xu *et al.*, 2019]. Motivated by sequence data modeling and contextual understanding of Transformers, some studies strive to construct Graph Transformers (GTs) to incorporate global nodes' semantic with local structural information. The self-attention mechanism from the vanilla Transformer calculates attention scores among tokens' pair and outputs the fusion of embeddings containing all attention information. This mechanism can be performed on the graph naturally, which tokens are considered as nodes and attention scores are seen as the importance of edges between two nodes. The process of aggregating different embeddings with attention weights also can be considered as a special case of applying the message-passing rule on a fully-connected graph. Some emerging GTs already achieve competitive or even surpassing performances against MPNNs in many graph tasks [Ying *et al.*, 2021] [Kreuzer *et al.*, 2021].

However, integrating Transformers and MPNNs to fully unleash their representational power on the large-scale graph data still faces some challenges. Firstly, the scheme is still ambiguous that injecting explicit connectivity information into Transformers to alleviate its deficiency in lack of strong inductive biases [Ma *et al.*, 2023]. Because the self-attention tends to ignore the structural information or relations between node pairs, it make Transformers hard to generate meaningful attention scores from the view of the graph topology information. Besides, for addressing these issues, those GTs with complex attention mechanism always lead to astonishing computational energy consumption. As we mentioned before, the vanilla self-attention mechanism is equivalent to calculate weights of every nodes' pair in a fully-connected graph. The computation and memory costs behind similar operations on the large-scale graph are unacceptable. Some approaches which attempt to improve the efficiency of GTs are still in early stages, and new effective solutions are yet to be explored further.

With development of neuromorphic hardwares, spiking neural networks (SNNs), as biologically plausible structures, have potential to break through the energy consumption bottleneck of Transformers. The brain-inspired architecture have attracted widespread attention as a new-generation efficient neural networks. Different from the neurons in artificial neural networks (ANNs), biological neurons interact with each other with sparse spikes [Eshraghian *et al.*, 2023]. Recently, the huge advancements in training algorithms enable SNNs composed by neurons to borrow sophisticated architectures from ANNs while further improve the energy efficiency of models. Some of them are increasingly becoming the cornerstone in different applications such as image recognition, gesture recognization and robot control [Roy *et al.*, 2019] [Zhou *et al.*, 2023]. Furthermore, it is also an effective way to utilize the sparse and binarized spikes output from biological neurons design a more lightweight, energy-efficient GNNs. Some pioneers already inject SNN into graph models and verify advantages of the fusion architecture on energy efficiency [Xu *et al.*, 2021] [Zhu *et al.*, 2022] [Li *et al.*, 2023]. The experimental results show promising potentiality of generalizing biologically plausible networks on the graph data, SNNs have been underappreciated and under-investigated in GTs. Introducing biological neurons into current Transformer frameworks may likely offer a sustainable, low-energy solution for GTs.

Based on SNNs, the sparse operations designed for spiking neurons bring huge reductions in memory and consumption costs. Simultaneously, as shown in Figure 1, we observe that replacing the softmax function with a spiking neuron, self-attention blocks may capture similar attention patterns. Due to the regularity of spikes, it is possible to learn attention patterns using spiking attention matrices from few time steps. These characteristics drive us to integrate the spiking neurons with GTs. In this study, we construct a spiking driven graph transformer (SGHormer). As far as we know, it is also the first methods which inject the spiking neurons into GTs. Specifically, there are two main components in SGHormer, spiking rectify block (SRB) and spiking graph self-attention (SGSA). SRBs recover and generate the approximated input embeddings, which can effectively alleviate the information loss during spiking. Based on observations, SGSA not only alleviates the problem about dependencies of SNNs on time steps, but also generates the spiking attention matrix in a power-efficient way. The contributions of this paper are summarized as follows:

The following instructions apply to submissions:

- We create an energy-saving graph transformer framework using biologically inspired spiking neurons.
- For this new spiking-driven GTs, We design Spiking Graph Attention Head (SGSA) and Spiking Rectify Blocks (SRB), which effectively utilize the inherent filer operation to simplify the self-attention calculation and alleviate the strong dependencies of spiking neurons on time steps.
- We compare our methods with 5 advanced GTs and 8 GNNs on PYG and OGB datasets. And we also compare the theoretical energy consumption of our models with that of other GTs. The results show that SGHormer can achieve comparable performances against other full-precision GTs with extremely low computational energy consumption.

2 Related Work

Spiking Neural Networks. Spiking Neural Networks (SNNs), characterized by low power consumption, event-driven features and biological plausibility, are considered the third generation of neural networks. Motivated by brain's neural circuitry, neurons in a SNN communicate with spikes which can be seen as electrical impulses when membrane potential reach the threshold. Early spiking neurons like Hodgkin-Huxley Model follow a biophysical mechanism that currents caused by action potentials will go through ion channels in the cell membrane [Gerstner *et al.*, 2014]. Starting from the Hodgkin-Huxley model, the derivation of simplified neuron models such as IF and LIF are proposed for adapting deep learning framework. Though output discrete, single-bit spikes extremely improve the efficiency of neural networks, binarized outputs also raise the non-differentiable problem which makes challenging for directly training SNNs through the backpropagation algorithm. ANN-to-SNN and surrogate gradients are two common ways to relieve the above questions [Cao *et al.*, 2015] [Zhou *et al.*, 2022]. There are lots of studies in the field of computer vision show that surrogate gradients can achieve approximated performances compared with the normal gradient decent.

Graph Transformers. As a transformative framework, Transformer and its variants achieve tremendous success and gradually become new benchmarks in various domains [Vaswani *et al.*, 2017] [Dosovitskiy *et al.*, 2021]. Due to self-attention mechanism can naturally be regarded as special case for importance calculation for nodes' pairs on the graph data. Emerged GTs verify this assumption. Some works focus on integrate the original MPNN with new Transformer framework [Rampášek *et al.*, 2022] [Ma *et al.*, 2023]. Other methods choose to further modify the attention calculation for reducing the quadratic complexity into linear complexity [Wu *et al.*, 2022] [Wei *et al.*, 2023]. In this work, we aim to build a spiking Graph Transformer architecture from the view of neuroscience and explore future Transformer-based neuromorphic chip design.

Positional/Structural encodings. Recent studies indicate that manually constructing positional and structural encodings (PE/SE) contributes to make standard MPNN more expressive than 1-Weisfeiler-Leman test [Srinivasan and Ribeiro, 2020]. There are two common encoding strategies, positional and structural encodings. Positional encodings is mainly used to generate embeddings that contain information about the location of nodes in the graph. There are many kinds of PE have been developed, for example, Laplacian PE or Weisfeiler-Lehman-based PE [Li *et al.*, 2020] [Dwivedi and Bresson, 2021]. The examples of structural encodings include degree of a node, random-walk SE and so on [Ying *et al.*, 2021] [Dwivedi *et al.*, 2022]. A common way to inject the supplementary information is adding or concatenating the positional or structural representations of the graph to with node features before the main Transformer model. In this work, we directly incorporate Laplacian PE and random-walk SE to generate auxiliary graph topological information.

3 Preliminaries

Spiking neural networks. Though the electrophysiological measurements can be calculated accurately by those complex conductance-based neurons, the complexity also limits widespread deployment in deep neural networks. Currently, most of SNNs consist of the simpler computational units, IF, LIF and PLIF [Gerstner *et al.*, 2014]. Specifically, a spiking neuron receives the weighted sum of input current and accumulates membrane potential. Subsequently, the neuron compared its membrane potential with a threshold to determine whether to generate the spikes. The membrane potential activity can be formulated as follows:

$$\tau \frac{dV^t}{dt} = -(V^t) + RI^t, \tag{1}$$

where V^t is the membrane potential, τ denotes a time constant of membrane and R denotes as the membrane resistance. For facilitating the deployment of biological neurons in deep learning, the membrane potential reset and spiking should be retained while relaxing the physically viable assumptions. To this end, Eq. 1 can be converted into an iterative expression as follows:

$$V^t = V^{t-1} + \beta(WX^t - (V^{t-1} - V_{reset})), \tag{2}$$

$$V^t = V^t(1 - S^t) + V_{reset}S^t, \tag{3}$$

$$S^t = \begin{cases} 1, & V^t \geq V_{th} \\ 0, & otherwise \end{cases}, \tag{4}$$

where β can be considered as simplified a decay constant. We utilize $snn(\cdot)$ to denote a LIF neuron hereafter. And we use the surrogate gradient to approximate gradients of parameters as follows:

$$\frac{\partial \mathcal{L}}{\partial S}\frac{\partial S}{\partial V}\frac{\partial V}{\partial I}\frac{\partial I}{\partial W} \approx \frac{\partial \mathcal{L}}{\partial S}\frac{\partial \tilde{S}}{\partial V}\frac{\partial V}{\partial I}\frac{\partial I}{\partial W}, \tag{5}$$

where $\tilde{S}$ is function substitution. In this work, we also employ spike gradients, which enables the model to reach the convergence quickly even with low time steps.

Self-attention. One of the most prominent components of Transformer is multi-head self-attention mechanism. For $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, and $X \in \mathbb{R}^{N \times d}$ denotes the nodes' features. let $X \in \mathbb{R}^{N \times d}$ be the input to a vanilla self-attention layer, where N is the number of nodes in a graph and d is dimensions of hidden embeddings. Query, key and value matrices are calculated by corresponding learnable projection matrices respectively, as defined below:

$$Q = XW_q, K = XW_k, V = XW_v, \tag{6}$$

where $W_q, W_k, W_v \in \mathbb{R}^{d \times d'}$. For Q and K, the layer calculates dot products of a query with all keys. Results are divided by $\sqrt{d'}$ and fed into a softmax function to calculate attention scores of each value. At last, for multi-head self-attention, the concatenation of output embeddings from M different heads will be integrated and applied a linear transformation:

$$H_m = softmax(\frac{Q_m K_m^T}{\sqrt{d'}})V_m, \tag{7}$$

$$H = Linear(H_1 \parallel H_2 \parallel ... \parallel H_M), \tag{8}$$

where $\parallel$ denotes concat operations.

4 Methodology

In this section, we detail a new spike-based transformer framework to change attention calculation into a graph reconstruction by spiking neurons. The framework of SGHormer is shown in Figure 2. As depicted in figure, SGHormer changes position encodings and corresponding attributes of nodes from continuous and full-precision values into rate coded spikes by rate-based encoder. Then spikes are send into the spiking graph self-attention block. SGSA binarizes global attention scores which turn the attention calculate task into a graph reconstruction task dominated by spiking neurons. Simultaneously, SGSA create the local embeddings of nodes using explicit connectivity information. At last, the sparse spiking embeddings are fed into a output head to generate the corresponding predictive results for downstream classification or regression tasks.

4.1 Rate-coded nodes' feature

As a rate-based SNN, we follow the same hypothesis that the spiking rate is proportional to the importance of patterns in nodes' features [Zhu *et al.*, 2022]. The higher intensity of features is equal to a higher spike count or spiking rate in the same time interval T. Let X_{feat} and X_{encode} be input node features and positional/structural encodings, respectively. The concatenation of above embedding X are fed into a rate encoder to generate rate-coded multi-temporal spikes $S = \{S^1, S^2, ..., S^T\}$. There are two common options to turn inputs into rate coding spikes. For a probability-based encoder, it consider the rate encoding as a Bernoulli trial. Another approach repeatedly pass the embeddings of nodes into a shared spiking neuron T times. We choose the latter as rate-based encoder in SGHormer. The process can be defined by the following formulations:

$$S_i^t = snn(X_i) = snn(Linear(X_{i,feat} \parallel X_{i,encode})), \tag{9}$$

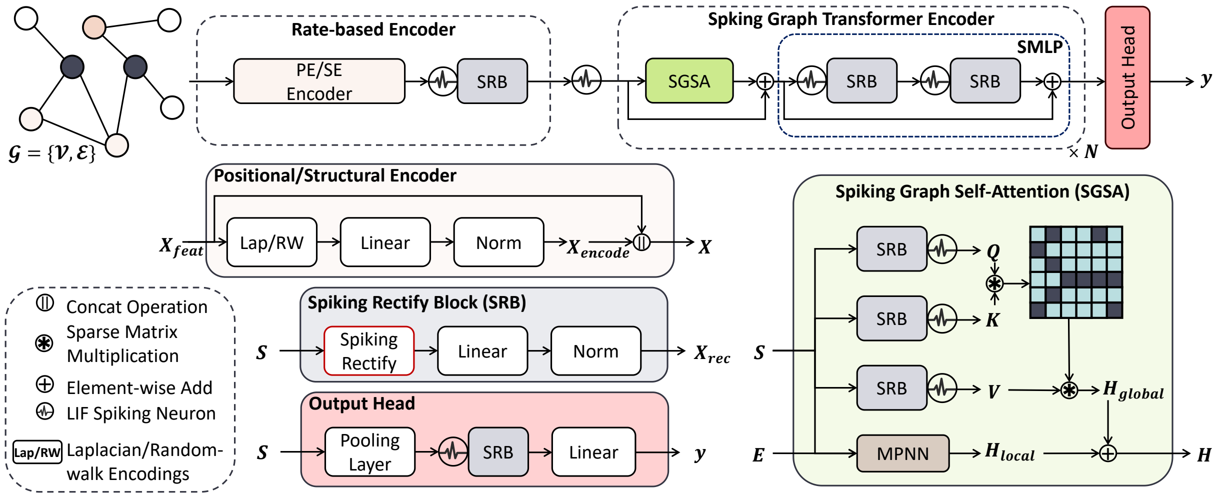

Figure 2: The framework of SGHormer

where X_i is the integrated embedding of node i, S_i^t denotes output spikes at the t-th time step. Calculating the spiking rate along T time steps can yield approximate estimate of raw embeddings before entering spiking neurons.

4.2 Spiking rectify block

As we mentioned before, all layers in SGHormer communicate with each other only using sparse and binarized spikes. It means that there are lots of operations involving the conversion from full-precision values to binary spikes. While some attempts demonstrate that constructing a deeper networks can also improve performances of SNNs [Fang *et al.*, 2021] [Zheng *et al.*, 2021] [Hu *et al.*, 2023], the negative impact of information loss becomes prominent on a more complex model and those advantages brought by deeper network framework will evaporate. Hence, we design a extra spiking rectify block (SRB) to recover input raw embeddings with real values from output spikes. We assume a raw embedding x_i follow the normal distribution. Based on the output spiking vector s_i, the mean $\hat{\mu}$ and variance $\hat{\sigma}$ of spikes across multiple time steps can be calculated. Subsequently, SRB utilizes the approximate results and rectifies output spikes filling values. The specific process can be formulated as follows:

$$\hat{\mu}_i = \frac{1}{T}\sum_{t=1}^{T} s_i^t, \hat{\sigma}_i = \frac{1}{T}\sum_{t=1}^{T}(s_i^t - \hat{\mu}_i)^2, \tag{10}$$

$$\hat{X}^t = S^t - W \odot U^t, U^{\sim}N(\hat{\mu}, 1-\hat{\sigma}), \tag{11}$$

$$X_{rec}^t = rec(S^t) = BN(Linear(\hat{X}^t), \tag{12}$$

where W^t is a learnable weight matrix, $\odot$ is the Hadamard product. As shown in Figure 2, SRB utilizes the correlation between output spikes and input embeddings to reconstruct nodes' embedding rather than apply linear transformation and normalization directly.

4.3 Spiking graph attention head

Different from the vanilla transformer, GTs not only need capture the local connectivity information utilizing explicit edges in the graph, but also calculate global attention scores to infer relation between each nodes' pair [Ying *et al.*, 2021] [Rampášek *et al.*, 2022]. Benefit from sparse and binarized spikes, spiking graph self-attention (SGSA) contain MPNN and self-attention blocks to undertake above two objectives with low computation and memory overhead. Specifically, we fed the query, key and value matrices into different spiking neurons. Output matrices after spiking neurons is sparse and binary, which only contain 0/1 elements. Therefore, the linear operation on these spiking matrices is only addition. The matrix multiplication in attention computation also can be transformed into a sparse form to further reduce memory consumption. On the basis of above sparse operations, SGSA generates node embeddings with local and global structure information. Besides, consistent with the original self-attention architecture, we construct a spiking multilayer perceptron (SMLP) to project outputs into the embedding space. The combination of above blocks can be written as:

$$H_{local}^l = MPNN(S^l, E), \tag{13}$$

$$\begin{aligned} H_{global}^l = \tilde{A}^l V^l &= \Theta(Q_{sp}^l, K_{sp}^l)V_{sp}^l \\ &= (g(Q^l) \circledast g(K^l))g(V^l) \end{aligned}, \tag{14}$$

$$S^{l+1} = SMLP(H_{local}^l + H_{global}^l), \tag{15}$$

where $g(\cdot) = snn(rec(\cdot))$, $\circledast$ denotes the sparse matrix multiplication. It also can be replaced by XNOR and bitcount operations [Rastegari *et al.*, 2016]. Notably, we just use a plain message passing layer to aggregate neighbors' and edges' feature, which can be defined as:

$$h_i = MPNN(x_i, e) = x_i + \sum_{j \in \mathcal{N}(i)} (wx_j + e_{ij}), \tag{16}$$

where $\mathcal{N}(\cdot)$ denotes the immediate neighborsof node, e_{ij} is the edge feature between nodes i and j. As shown in Eq. 14, The global attention scores are calculated by query and key

Model	ZINC	MNIST	CIFAR10	PATTERN	CLUSTER
	MAE↓	Accuracy↑	Accuracy↑	Accuracy↑	Accuracy↑
GCN	0.367±0.011	90.705±0.218	55.710±0.381	71.892±0.334	68.498±0.976
GIN	0.526±0.051	96.485±0.252	55.255±1.527	85.387±0.136	64.716±1.553
GAT	0.384±0.007	95.535±0.205	64.223±0.455	78.271±0.186	70.587±0.447
GatedGCN	0.282±0.015	97.340±0.143	67.312±0.311	85.568±0.088	73.840±0.326
GatedGCN-LSPE	0.090±0.001	-	-	-	-
PNA	0.188±0.004	97.940±0.12	70.350±0.630	-	-
DGN	0.168±0.003	-	**72.838±0.417**	86.680±0.034	-
GSN	0.101±0.010	-	-	-	-
SAN	0.139±0.006	-	-	86.581±0.037	76.691±0.650
Graphormer	0.122±0.006	-	-	-	-
K-Subgraph SAT	0.094±0.008	-	-	**86.848±0.037**	77.856±0.104
EGT	0.108±0.009	98.051±0.126	68.702±0.409	86.821±0.020	**79.232±0.348**
GPS	**0.070±0.004**	**98.173±0.087**	72.298±0.356	86.685±0.059	78.016±0.180
ours	0.117±0.032	96.850±0.247	67.740±0.158	86.527±0.620	71.279±0.205

Table 1: Test performances on five Benchmarking-GNNs datasets. The best result are highlighted in **red**. Color blocks represent the difference between SGHormer and other full-precision methods, with darker colors indicating larger performance gaps.

Model	ogbg-molhiv	ogbg-molpcba
	AUC↑	AP↑
GCN+virtual node	75.90±1.1	24.24±0.3
GIN+virtual node	77.07±1.4	27.03±0.2
GatedGCN-LSPE	-	26.70±0.2
PNA	79.05±1.3	28.38±0.3
DeeperGCN	78.58±1.1	27.81±0.3
DGN	79.70±0.9	28.85±0.3
GSN (directional)	**80.39±0.9**	-
SAN	77.85±0.2	27.65±0.4
GraphTrans (GCN-Virtual)	-	27.61±0.2
K-Subtree SAT	-	-
GPS	78.80±1.0	**29.07±0.2**
SGHormer	77.47±0.4	27.43±0.1

Table 2: Test performances on two OGB datasets.

spiking matrices. From this perspective, the task about inferring latent relations on a fully-connected graph can be considered as the graph reconstruction controlled by spiking neurons. Because the nonnegativity of spikes, we further remove the softmax function to simplify the computation of self-attention. For any given self-attention layer, a simple way to compute the global embedding $H = \{H^1, H^2, ..., H^T\}$ for T time steps can be written as:

$$H^1 = \tilde{A}^1 V^1, H^2 = \tilde{A}^2 V^2, ..., H^T = \tilde{A}^T V^T, \quad (17)$$

However, we believe that such operations overlook the inherent filter operations of SNNs. And the arrival times of spikes can also reveal the intensity between different elements using the rate-coding method mentioned in the previous section. The earlier a spike arrives, the greater its input value compared to the membrane potential threshold of a spiking neuron. Therefore, we consider an extreme scenario that the most important latent relations emerges in $\tilde{A}^{1,l}$ after filtering by spiking neuron. At this point, the process of generating global embedding can be updated as:

$$H^1 = \tilde{A}^1 V^1, H^2 = \tilde{A}^1 V^2, ..., H^T = \tilde{A}^1 V^T, \quad (18)$$

$$\tilde{A}^1 = g(Q^1) \circledast g(K^1)), \quad (19)$$

This simplified operation significantly reduces the dependence of SNNs on long time steps. Besides, it still ensuring that the global connectivity information can be captured by SGSA. We choose the latter one in our experiments.

5 Experiments

5.1 Graph classification

In this section, we compare SGHormer against various message passing neural networks and graph transformers on Benchmarking-GNNs [Dwivedi *et al.*, 2023] and OGB [Hu *et al.*, 2021]. For comprehensively validating the effectiveness of SGHormer, selected datasets covering various graph-related tasks such as graph regression, graph classification and node classification. All experiments are conducted on the standard splits of the evaluated datasets. We perform our model on each dataset 5 times with different random seeds to report the mean and standard deviation. All above experiments are conducted on a single NVIDIA RTX 3090 GPU if not explicitly stated otherwise.

Datasets. We select five datasets including ZINC, MNIST, CIFAR10, PATTERN and CLUSTER from Benchmarking-GNNs to evaluate our method. For the open graph benchmark (OGB), we select two molecular property prediction datasets with different scales, namely ogbg-molhiv and ogbg-molpcba.

Baselines. As of now, the spiking-related graph transformers or any GNNs used for graph-level tasks has not been located. Therefore, all selected baselines are in full-precision

Model	ZINC			MNIST			PATTERN			ogbg-molhiv		
	Param (MB)	Mem (GB)	Eng (mJ)	Param (MB)	Mem (GB)	Eng (mJ)	Param (MB)	Mem (GB)	Eng (mJ)	Param (MB)	Mem (GB)	Eng (mJ)
SAN	0.19	0.58	22.73	0.48	15.07	200.43	0.39	14.68	907.49	0.49	6.45	40.44
Graphormer	0.10	0.41	6.64	0.15	2.71	51.81	0.10	0.51	212.18	0.15	2.76	11.79
SAT	0.24	0.52	13.65	0.36	17.21	112.99	0.24	2.97	599.0	0.37	2.05	23.38
GPS	0.16	0.40	6.88	0.32	15.44	52.55	0.21	0.73	214.63	0.32	2.07	12.17
Average	0.17	0.48	12.47	0.33	12.61	104.44	0.24	4.72	483.33	0.33	3.33	21.94
SGHormer	0.09	0.56	0.18	0.13	6.46	0.58	0.08	1.00	1.64	0.14	4.30	0.30

Table 3: The parameter size (MB), memory usage (GB) and theoretical energy consumption (mJ) of various GTs. Color blocks represent the improvement in efficiency compared SGHormer with other full-precision methods, with darker colors indicating larger disparities.

form. One of the main categories among these models are message passing neural networks: GCN [Kipf and Welling, 2017], GIN [Xu *et al.*, 2019], GAT [Veličković *et al.*, 2017], GatedGCN [Bresson and Laurent, 2018], GatedGCN-LSPE [Dwivedi *et al.*, 2022], PNA [Corso *et al.*, 2020], DGN [Beaini *et al.*, 2021], GSN [Bouritsas *et al.*, 2023]. The others are some advanced graph transformers employed in graph-level tasks widely: SAN [Kreuzer *et al.*, 2021], Graphormer [Ying *et al.*, 2021], SAT [Chen *et al.*, 2022], EGT [Hussain *et al.*, 2022], GPS [Rampášek *et al.*, 2022].

Overall performance. The comparative results are demonstrated in Table 1. It is evident that, despite some gaps compared to state-of-the-art methods based on binary ground truth distances, SGHormer has achieved comparable predictive performance on most datasets through carefully designed network structures. In certain datasets like ZINC, it even outperforms the predictions of full-precision ground truth methods. For message passing neural networks, SGHormer outperforms GCN, GAT and GIN on every datasets. Since the current model only employs a plain MPNN and a simplifying global attention mechanism, its performance remains subpar on datasets with plain node's feature (eg. one-hot encoding) like CLUSTER. Improving performance on such datasets is one of the future directions of our work.

5.2 Theoretical Energy Consumption

To examine the energy efficiency of SGHormer, we measure SGHormer and other GTs from three different metrics, model size, memory usage and theoretical energy consumption. However, directly applying the model on neuromorphic chip is rarely explored [Zhu *et al.*, 2022]. To investigate the energy consumption of SGHormer, we derived the theoretical energy consumption from previous works [Zhou *et al.*, 2022]. Since GTs are still in the early stages of development, there is no standard model size set for each task. For the sake of fairness in comparison, we set the same fixed hyperparameters including the number of layers, the number of heads, the dimension of hidden embeddings for each model while calculating the energy consumption. Besides, different GTs employing various encoding strategies, part of methods preprocess structural information and embed it into node features, while others tend to build a learnable encoder block into the network. We only calculate the energy consumption of transformer encoder in inference step by counting floating point operations (FLOPs) and synaptic operations (SOPs). And the theoretical energy consumption of SGHormer can be formulated as follows:

$$E = E_{coding} + \sum_{l=1}^{L} E_{trans} \tag{20}$$

$$\begin{aligned} E = {} & \alpha_f FLOP_{coding} \\ & + \alpha_s \sum_{t=1}^{T}\sum_{l=1}^{L}(SOP^{t,l}_{srb} + SOP^{t,l}_{mpnn} + SOP^{t,l}_{attn}) \end{aligned} \tag{21}$$

$$SOP^{t,l} = r^{t,l} \times FLOP^{t,l} \tag{22}$$

where α_f and α_s, as scale factors for floating point and synaptic operations, which are set to 4.5 and 0.9, respectively. $r^{t,l}$ is fire rate of block in the l-th layer at the t-th time step. The theoretical energy consumption results are shown in Table 3. For all datasets, our method are obviously outperform than the other GTs in size and energy consumption. The average energy consumption is 153x lower compared to other models. This advantage is more pronounced on MNIST and PATTERN which contain more nodes and edges. Besides, the size of our model is smaller which contributes to extend SGHormer to edge devices. Currently, there is still a lack of relevant operations and support for sparse and binarized spikes, making it challenging to demonstrate the model's advantages in terms of memory usage. We believe that further designing customized operators for spiking neural network (SNN) on GPUs will accelerate the development of SNNs.

5.3 Component analysis

To elaborately discuss the effectiveness of different components, we construct a series of ablation studies on the SGHormer. Specifically, the experiments primarily assess the impacts of three components including: spiking rectify block, spiking graph self-attention and spiking neuron. There are four comparative methods implemented by removing or replacing one of components. The results are demonstrated in the Table 4

Model	ZINC(↓)	PATTERN(↑)	ogbg-molhiv(↑)
ours	0.117	**86.527**	77.473
- SRB	0.129	85.803	77.241
+ SATT	0.114	86.518	76.004
+ IF	0.126	86.179	76.456
+ PLIF	**0.109**	86.267	**77.478**

Table 4: Ablation studies on SGHormer. $-x$ means removing the component x from SGHormer. And $+x$ means replacing the original component in SGHormer with x.

Spiking rectify block. As shown in table 4, removing the spiking rectify block significantly impair predictive performances of SGHormer on three datasets. For a deep spiking neural network, the inputs will become sparser after each layer without any intermediate processing or extra supplementary information. In addition, it bring tons of quantization error that encoding full-precision raw data into binarized spikes. Some existing SNNs are sill strongly dependent on the imprecise spiking representations and directly pass spiking outputs to subsequent processes. We suggest that the limitation make SNNs hard to develop similar the network structure with a vast number of parameters like ANNs. SRBs consider received spikes as a biased data, the blocks attempt to learn a estimator for roughly recovering raw embedding.

Spiking graph self-attention. What have been discussed in the previous section is that output spikes of layers follows some certain regulation along T time steps. For those raw inputs that far exceed the threshold of membrane potential, they often emit early and with a higher firing rate under the straightforward rate-based encoder. Because of the regularity of output spikes, we suggest that just using few spiking attention matrices from certain time steps can approximatively capture the essential graph structure information. This assumption can be verified in Table 4. The self-attention through time described in Eq. 17 (hereafter, SATT) can't provide more valuable relation information among nodes compare with the component which just uses the spiking attention matrix at 1-st the time step.

Spiking neuron. Except for the two core components in the SGHormer, we also explore the influences of spiking neurons. Table 4 shows that the predictive performances of PLIF with learnable membrane time constants surpass surpass that of LIF and IF. Due to the spiking attention matrix are control the by corresponding spiking neurons. Currently, There is still no a implementation of a common neuron that can be generalized to the different graph data. We believe that developing and designing specialized spiking neurons for GTs or graph-related tasks may further improve the performance of SGHormer.

5.4 Parameters analysis

For SGHormer, there are several critical hyperparameters like the membrane potential threshold V_{th} and number of time steps T, will directly affect the performances of SGHormer. In this section, we deploy experiments on these parameters

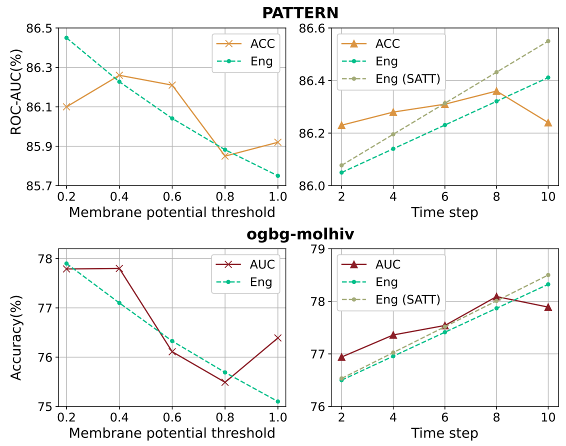

Figure 3: Performances of SGHormer with different membrane potential threshold and time step on PATTERN and ogbg-molhiv datasets.

to explore the correlations between predictive performances and energy consumption of model with different parameter settings. The results are depicted in Figure 3. We visualize the changes of metrics as parameters change and scale the theoretical energy consumption to a similar interval. For the membrane potential threshold, it is directly connected with the spiking rate of whole model. As the threshold increases, the spiking rate of spiking neurons and energy consumption both decreases. We observe that optimal choices of threshold is quite different on different datasets. This is one of reasons why some parameterized LIF, which can automatically adjust membrane-related parameters, may achieve better performances. The number of time steps is essential to approximate the real-valued inputs in rate-based SNNs. Theoretically, the firing rate can represent the original real value accurately as the number of time steps goes to infinity. We observe that The advantage of increasing the time steps for SGHormer is more in terms of speed of convergence rather than its performance. It make SGHormer can achieve good performance with few time steps while maintain a low-level energy consumption. Compared with SATT, SGHormer is more energy-efficient on large-scale graph data due to the simplifying attention computation.

6 Conclusion

In this study, we have explored a energy-saving graph transformer driven by SNN. In order to create the fusion of GT and SNN, we design spiking rectify block (SRB) and spiking graph self-attention (SGSA). SRB enables SGHormer to maintain representation power while let spikes as the communication signals among layers. And SGSA partly alleviates internal drawbacks of SNNs on strong dependencies of time steps during calculating the attention scores. Massive experiments conducted to on graph-level benchmarks show that well-designed spiking-based GTs can bridge the performance gaps and achieve the comparable performances with extremely low energy consumption. As a energy-saving solutions from the perspective of the biological structure, SGHormer have potential to pave the path for deploying GTs on edge devices.

References

[Beaini *et al.*, 2021] Dominique Beaini, Saro Passaro, Vincent Létourneau, Will Hamilton, Gabriele Corso, and Pietro Lió. Directional graph networks. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 748–758. PMLR, 18–24 Jul 2021.

[Bouritsas *et al.*, 2023] Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M. Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):657–668, 2023.

[Bresson and Laurent, 2018] Xavier Bresson and Thomas Laurent. Residual gated graph convnets, 2018.

[Cao *et al.*, 2015] Yongqiang Cao, Yang Chen, and Deepak Khosla. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 113:54–66, 2015.

[Chen *et al.*, 2022] Dexiong Chen, Leslie O'Bray, and Karsten Borgwardt. Structure-aware transformer for graph representation learning. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 3469–3489. PMLR, 17–23 Jul 2022.

[Corso *et al.*, 2020] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 13260–13271. Curran Associates, Inc., 2020.

[Dai *et al.*, 2018] Hanjun Dai, Zornitsa Kozareva, Bo Dai, Alex Smola, and Le Song. Learning steady-states of iterative algorithms over graphs. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1106–1114. PMLR, 10–15 Jul 2018.

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

[Dwivedi and Bresson, 2021] Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs, 2021.

[Dwivedi *et al.*, 2022] Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations, 2022.

[Dwivedi *et al.*, 2023] Vijay Prakash Dwivedi, Chaitanya K. Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24(43):1–48, 2023.

[Eshraghian *et al.*, 2023] Jason K. Eshraghian, Max Ward, Emre O. Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D. Lu. Training spiking neural networks using lessons from deep learning. *Proceedings of the IEEE*, 111(9):1016–1054, 2023.

[Fang *et al.*, 2021] Wei Fang, Zhaofei Yu, Yanqi Chen, Tiejun Huang, Timothée Masquelier, and Yonghong Tian. Deep residual learning in spiking neural networks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 21056–21069. Curran Associates, Inc., 2021.

[Gerstner *et al.*, 2014] Wulfram Gerstner, Werner M Kistler, Richard Naud, and Liam Paninski. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

[Hamilton *et al.*, 2018] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs, 2018.

[Hu *et al.*, 2021] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs, 2021.

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

[Hussain *et al.*, 2022] Md Shamim Hussain, Mohammed J. Zaki, and Dharmashankar Subramanian. Global self-attention as a replacement for graph convolution. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, page 655–665, New York, NY, USA, 2022. Association for Computing Machinery.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017.

[Kreuzer *et al.*, 2021] Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 21618–21629. Curran Associates, Inc., 2021.

[Li *et al.*, 2020] Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. Distance encoding: Design provably more powerful neural networks for graph representation learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 4465–4478. Curran Associates, Inc., 2020.

[Li *et al.*, 2023] Jintang Li, Zhouxin Yu, Zulun Zhu, Liang Chen, Qi Yu, Zibin Zheng, Sheng Tian, Ruofan Wu, and Changhua Meng. Scaling up dynamic graph representation learning via spiking neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(7):8588–8596, Jun. 2023.

[Ma *et al.*, 2023] Liheng Ma, Chen Lin, Derek Lim, Adriana Romero-Soriano, Puneet K. Dokania, Mark Coates, Philip Torr, and Ser-Nam Lim. Graph inductive biases in transformers without message passing, 2023.

[Rampášek *et al.*, 2022] Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 14501–14515. Curran Associates, Inc., 2022.

[Rastegari *et al.*, 2016] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 525–542, Cham, 2016. Springer International Publishing.

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

[Srinivasan and Ribeiro, 2020] Balasubramaniam Srinivasan and Bruno Ribeiro. On the equivalence between positional node embeddings and structural graph representations, 2020.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[Veličković *et al.*, 2017] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[Wei *et al.*, 2023] Yinwei Wei, Wenqi Liu, Fan Liu, Xiang Wang, Liqiang Nie, and Tat-Seng Chua. Lightgt: A light graph transformer for multimedia recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, page 1508–1517, New York, NY, USA, 2023. Association for Computing Machinery.

[Wu *et al.*, 2022] Qitian Wu, Wentao Zhao, Zenan Li, David P Wipf, and Junchi Yan. Nodeformer: A scalable graph structure learning transformer for node classification. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 27387–27401. Curran Associates, Inc., 2022.

[Xu *et al.*, 2019] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks?, 2019.

[Xu *et al.*, 2021] Mingkun Xu, Yujie Wu, Lei Deng, Faqiang Liu, Guoqi Li, and Jing Pei. Exploiting spiking dynamics with spatial-temporal feature normalization in graph learning, 2021.

[Ying *et al.*, 2018] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, page 974–983, New York, NY, USA, 2018. Association for Computing Machinery.

[Ying *et al.*, 2021] 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? In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 28877–28888. Curran Associates, Inc., 2021.

[Zheng *et al.*, 2021] Hanle Zheng, Yujie Wu, Lei Deng, Yifan Hu, and Guoqi Li. Going deeper with directly-trained larger spiking neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12):11062–11070, May 2021.

[Zhou *et al.*, 2022] Zhaokun Zhou, Yuesheng Zhu, Chao He, Yaowei Wang, Shuicheng Yan, Yonghong Tian, and Li Yuan. Spikformer: When spiking neural network meets transformer, 2022.

[Zhou *et al.*, 2023] Chenlin Zhou, Liutao Yu, Zhaokun Zhou, Zhengyu Ma, Han Zhang, Huihui Zhou, and Yonghong Tian. Spikingformer: Spike-driven residual learning for transformer-based spiking neural network, 2023.

[Zhu *et al.*, 2022] Zulun Zhu, Jiaying Peng, Jintang Li, Liang Chen, Qi Yu, and Siqiang Luo. Spiking graph convolutional networks, 2022.

[Zhu *et al.*, 2023] Yuchang Zhu, Jintang Li, Liang Chen, and Zibin Zheng. The devil is in the data: Learning fair graph neural networks via partial knowledge distillation, 2023.