Dynamic Reactive Spiking Graph Neural Network

Han Zhao[1], **Xu Yang**[1] *, **Cheng Deng**[1] *, **Junchi Yan**[2]

[1] School of Electronic Engineering, Xidian University, China
[2] Department of Computer Science and Engineering & MoE Key Lab of AI, Shanghai Jiao Tong University, China
{hzhao1698, xuyang.xd, chdeng.xd}@gmail.com, yanjunchi@sjtu.edu.cn

Abstract

Spiking Graph Neural Networks are emerging tools for analyzing graph data along with low energy consumption and certain biological fidelity. Existing methods directly integrate same-reactive spiking neurons into graph neural networks for processing propagated graphs. However, such same-reactive neurons are not biological-functionality enough compared to the brain's dynamic-reactive ones, limiting the model's expression. Meanwhile, insufficient long-range neighbor information can be excavated with the few-step propagated graph, restricting discrimination of graph spiking embeddings. Inspired by the dynamic cognition in the brain, we propose a Dynamic Reactive Spiking Graph Neural Network that can enhance model's expressive ability in higher biological fidelity. Specifically, we design dynamic reactive spiking neurons to process spiking graph inputs, which have unique optimizable thresholds to spontaneously explore dynamic reactive states between neurons. Moreover, discriminative graph positional spikes are learned and integrated adaptively into spiking outputs through our neurons, thereby exploring long-range neighbors more thoroughly. Finally, with the dynamic reactive mechanism and learnable positional integration, we can obtain a powerful and highly bio-fidelity model with low energy consumption. Experiments on various domain-related datasets can demonstrate the effectiveness of our model. Our code is available at https://github.com/hzhao98/DRSGNN.

Introduction

Graph neural networks (GNNs), reconciling the expressive power of graphs with the learning capacity of deep neural networks, have become powerful tools for analyzing ubiquitous graph data. During training and inference, existing GNNs (Defferrard, Bresson, and Vandergheynst 2016; Kipf and Welling 2016; Chen et al. 2020; Yang et al. 2020b,a) commonly update graph features by transforming and aggregating topological neighbors under persistently active neurons. Unfortunately, such persistently active neurons would consume high energy (Anthony, Kanding, and Selvan 2020); meanwhile, GNNs have less biological fidelity due to non-compliance with biological neurons' operating mechanism.

Recently, a novel generation of neural networks–Spiking Neural Networks (SNNs) (Bellec et al. 2018; Cheng et al. 2020; Rathi and Roy 2020; Fang et al. 2021; Wang, Cheng, and Lim 2022; Lee, Delbruck, and Pfeiffer 2016; Wu et al. 2018; Jin, Zhang, and Li 2018; Neftci, Mostafa, and Zenke 2019), has emerged to simulate intermittent activities and spiking communications in biological neurons, showing the potential of low energy consumption and high biological fidelity. Thus, to enable GNNs to satisfy the low-energy and high bio-fidelity, studying Spiking GNNs is necessary on the ubiquitous large-scale graph data. Existing methods (Xu et al. 2021; Wang and Jiang 2022; Zhu et al. 2022) directly replace the conventional GNNs' neurons with spiking ones for processing the propagated graph, where these spiking neurons have consistent firing thresholds with the same reactive state. According to the dynamic cognition observation that biological neurons have dynamic reactive states on processing signals (Deco, Cruzat, and Kringelbach 2019; Deco, Vidaurre, and Kringelbach 2021), such same-reactive neurons may have not enough capacity in terms of biological functionality, and thus limit the expressive ability of the model. In addition, insufficient long-range neighbor information can be captured with the few-step propagated graph, leading to non-discriminative spiking graph results.

To address the above problems, we propose a Dynamic Reactive Spiking Graph Neural Network that can sustain high biological fidelity and efficiently capture long-range neighbors. Under the guidance of the brain's dynamic cognition, each neuron of our model would generate a unique reactive state according to its optimizable threshold. During optimization, the reactive dynamics between neurons can be spontaneously explored for more biological fidelity, thus generating more discriminative spiking outputs. In addition, the graph positional spikes are learned and integrated adaptively into spiking outputs via our neurons, thereby excavating long-range neighbors more thoroughly. With the dynamic reactive mechanism and learnable positional integration, the bio-fidelity and expressive ability of our model can finally be enhanced. **The highlights of our work are:**

1) We propose a dynamic reactive spiking graph neural network that has powerful learning capability in high biological fidelity. 2) Guided by the reactive cognition in the brain, we design dynamic reactive spiking neuron models with unique optimizable thresholds for spontaneously exploring the neurons' reactive dynamics. 3) To sufficiently excavate long-range neighbors and thus enhance the mod-

*The corresponding authors

el expression, we integrate learnable positional spikes adaptively into spiking graph outputs. 4) Extensive experiments can demonstrate the powerful learning capability and low energy consumption of our proposed model.

Related Work

Spiking Neural Networks

Training Strategies. Existing deep SNNs are mainly divided into artificial neural networks (ANNs)-converted and direct-training SNNs. The former (Cao, Chen, and Khosla 2015; Diehl et al. 2015; Rueckauer et al. 2017) provide an alternative ANNs training strategy for SNNs, to further utilize their low energy consumption on developed deep learning studies. The latter ones (Lee, Delbruck, and Pfeiffer 2016; Jin, Zhang, and Li 2018; Neftci, Mostafa, and Zenke 2019; Kheradpisheh and Masquelier 2020) perform direct and efficient training through the ANNs' error backpropagation paradigm, which commonly use a continuous function to approximate the spiking function or its derivative.

Spiking Neuron Models. The spiking neuron model, inspired by neuroscience, is the basic unit and plays a vital role in controlling data transmission in SNNs. A classic LIF model (Gerstner and Kistler 2002) simulates the generation mechanism of the action potential in a simplified way. On this basis, a series of variant methods have been studied (Fourcaud-Trocmé et al. 2003; Brette and Gerstner 2005; Bellec et al. 2018; Cheng et al. 2020; Rathi and Roy 2020; Wang, Cheng, and Lim 2022). For example, inspired by Lateral Interactions (Ratliff, Hartline, and Lange 1974) in neuroscience, LISNN (Cheng et al. 2020) integrates adjacent neurons' lateral into spiking neuron membrane potential mechanism. Several approaches (Rathi and Roy 2020; Wang, Cheng, and Lim 2022) attempt to explore dynamic reactive neural layers by optimizing the unique threshold at the layer level. Regarding the layer-level region as the unit area, such layer-level optimization methods default that all neurons in the same brain unit area have the same reactive state. However, recent brain's cognition researches (Deco, Cruzat, and Kringelbach 2019; Deco, Vidaurre, and Kringelbach 2021) observe that the neurons' reactive states may unevenly vary even in a tiny spatial brain area. Therefore, these layer-level optimization methods are not biological-functionality enough, which cannot sufficiently capture the reactive dynamics and thus limit the expressive ability of models. In our work, we design a neuron-level threshold optimization method to imitate the reactive dynamics between neurons in the brain, thus enhancing the model expression.

Graph Neural Networks

The pioneering works that apply neural networks to graphs (Gori, Monfardini, and Scarselli 2005; Scarselli et al. 2008; Bruna et al. 2013) learn node representation with recurrent neural networks or their improvements. Afterward, plenty of methods (Kipf and Welling 2016; Veličković et al. 2017; Wu et al. 2019; Liu, Gao, and Ji 2020; Dwivedi et al. 2022; Cui et al. 2022; Zeng et al. 2021; Luo et al. 2023) have been proposed to make GNNs more powerful. For example, Kipf. *et. al.* introduced a linear function to the spectral filter (Kipf and Welling 2016) to prevent performance degradation. Then, Wu *et. al.* treated neighborhood aggregation as a pre-computing process in SGC (Wu et al. 2019). Aiming to alleviate the performance-dropping problem in deep-layer conditions, Liu *et. al.* (Liu, Gao, and Ji 2020) proposed DAGNN to decouple the representation transformation and propagation to learn graph representations from larger receptive fields. And other studies (Kreuzer et al. 2021; Dwivedi et al. 2022; Cui et al. 2022) mainly utilize positional encodings, that play a central role in the most prominent neural networks (LeCun et al. 1998; Hochreiter and Schmidhuber 1997; Vaswani et al. 2017), to alleviate over-smoothing and over-squashing problems.

Recently, several models belonging to SNNs (Xu et al. 2021; Wang and Jiang 2022; Zhu et al. 2022) have been studied with low energy consumption and high bio-fidelity. They directly utilize the same-reactive spiking neurons (Gerstner and Kistler 2002) to process the propagated graph data with maintaining a low energy consumption and certain bio-fidelity. However, due to the dynamic cognition in the brain (Deco, Cruzat, and Kringelbach 2019; Deco, Vidaurre, and Kringelbach 2021), adopting the same-reactive neuron is not biological-functionality enough and would limit the model's expressive ability. Meanwhile, the few-step propagated graph cannot thoroughly excavate long-range neighborhood information, leading to non-discriminative spiking results. On the contrary, through dynamic reactive neurons and spiking positional integration, our proposed model can sustain high biological fidelity and efficiently excavate long-range neighbors.

Preliminaries

In this section, we provide the basic definition of graph and recap the preliminaries in GNNs and SNNs. We denote a graph as $G = \{\mathcal{V}, \mathbf{X}, \mathbf{A}, \mathbf{Y}\}$, where $\mathcal{V} = \{v_k\}_{k=1}^{n}$ is the set of n nodes, $\mathbf{X} \in \mathbb{R}^{n\times d}$ represents node features, and each node v_k is associated with a d-dimensional feature vector $\boldsymbol{x}_k \in \mathbb{R}^d$. $\mathbf{A} \in \mathbb{R}^{n\times n}$ is the intrinsic adjacent matrix, and $\mathbf{D}$ is the diagonal degree matrix of $\mathbf{A}$. $\mathbf{Y} \in \mathbb{R}^{n\times c}$ is the one-hot label matrix with c classes.

Graph Neural Networks

The graph convolution mechanism in our model is based on the previous work Simplifying Graph Convolutional Network (SGC) (Wu et al. 2019), which treats the neighborhood aggregation as a pre-computing process to reduce the excess complexity caused by collapsing weight matrices in regular GNNs. SGC first performs K-step graph feature propagation with the adjacent matrix, then employs a collapsing replacement of GNNs' resulting function – a single linear transformation on the propagated graph features. The operation of SGC with two-step feature propagation is:

$$\hat{\mathbf{X}} = \hat{\mathbf{A}}(\hat{\mathbf{A}}\mathbf{X}), \tag{1}$$

$$\mathbf{Y} = \hat{\mathbf{X}}\mathbf{W}, \tag{2}$$

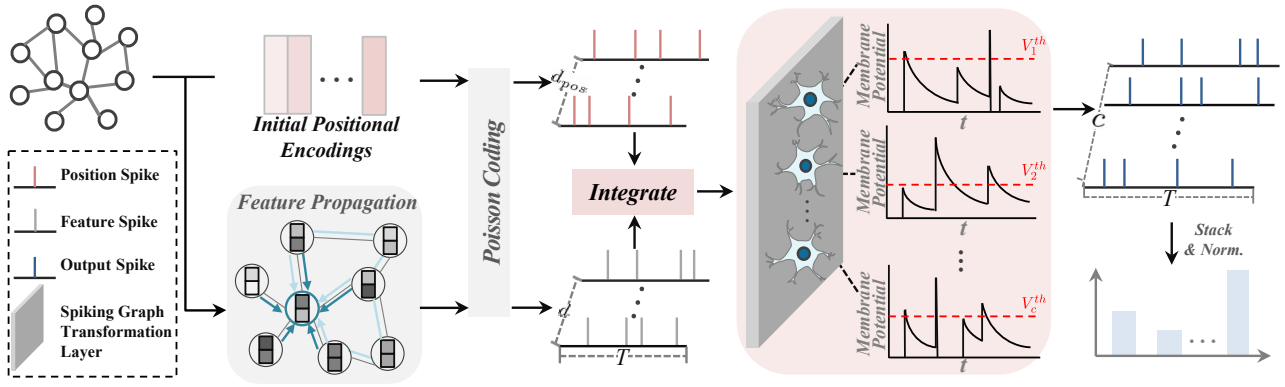

Figure 1: The illustration of our proposed model. We first utilize feature propagation and random walk to obtain the propagated graph features and initial positional features. Then, both features are converted into the spiking signals through Poisson coding, and integrated as the spiking inputs of our dynamic reactive spiking graph transformation (DRSGT) layer. Stimulated by these spiking inputs at t-timestep, each neuron in the DRSGT layer would reset and charge the membrane potential, and fire spikes according to its unique threshold, finally generating spiking outputs. During training, each neuron's unique threshold is optimized for spontaneously exploring reactive dynamics between neurons, thereby simulating the dynamic cognition in the brain. Meanwhile, the discriminative graph positional spikes can be learned and integrated adaptively into the spiking graph outputs, so as to sufficiently excavate the long-range neighbor information.

where $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$ is a renormalization operation on the feature propagation process with $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_n$. $\tilde{\mathbf{D}}$ is the degree matrix of $\tilde{\mathbf{A}}$, and $\mathbf{I}_n$ is the identity matrix.

Spiking Neural Networks

The Spiking Neuron model adopted as our base one is the Leakey Integrate-and-fire (LIF) (Gerstner and Kistler 2002). LIF imitates the action potential generation mechanism in the brain, that is, the neuron would fire a pulse when the membrane potential reaches the threshold V_{th}, and then the membrane potential would be resetted. The LIF model operated on the discrete domain can be described as:

$$V_i^t = \lambda V_i^{t-1} + \sum_j W_{ij} Z_j^t - V^{th} Z_i^{t-1}, \tag{3}$$

$$Z_i^{t-1} = \begin{cases} 1, & \text{if} \quad V_{i(rl)}^{t-1} > 1 \\ 0, & \text{otherwise} \end{cases}, V_{i(rl)}^{t-1} = \frac{V_i^{t-1}}{V^{th}}, \tag{4}$$

where V_i^{t-1} and Z_i^{t-1} represent the membrane potential value and spiking output value of the i-th post-neuron at $(t-1)$-th timestep, respectively. Z_j^t represents the spiking output value of the j-th pre-neuron at t-th timestep. $\lambda (0 \leq \lambda \leq 1)$ and W_{ij} are the leak factor and weight value connecting j-th pre-neuron and i-th post-neuron, respectively. V^{th} is the threshold value of the LIF neuron model. The neuron model is expected to process serial spiking inputs with T-duration. $V_{i(rl)}^{t-1}$ is the membrane potential value relative to V^{th} of the i-th post-neuron at $(t-1)$-th timestep. The first two terms of Equation 3 represent the potential leakage and accumulation of neurons, respectively, while Equation 4 is the firing process when the potential reaches the threshold. And the last term of Equation 3 is the membrane potential reset; that is, after firing a spike, the neuron would reset its potential.

Dynamic Reactive Spiking GNN

In this section, we will introduce the dynamic reactive spiking graph neural network guided by the brain's dynamic cognition, whose framework is depicted in Figure 1.

To keep to the high training efficiency properties of SNNs, the feature propagation is also regarded as the pre-computing process in our model, then the spiking linear graph transformation is employed on such propagated-graph feature information.

Input Preprocessing for Spiking GNN

Since the spiking neural network should be stimulated by spiking sequential input data, we utilize the common strategy - Poisson coding to code $\hat{\mathbf{X}}$ into spiking inputs with T duration, which generates random values and compares them with the float values in $\hat{\mathbf{X}}$ to produce the spiking inputs at each timestep. The probability distribution of the spiking number is subject to Poisson distribution. We denote the spiking form of the propagated-feature matrix $\hat{\mathbf{X}}$ at t-th timestep as $\hat{\mathbf{X}}^t$. In this way, the spiking sequential propagated-graph features $[\hat{\mathbf{X}}^1, \hat{\mathbf{X}}^2, \cdots, \hat{\mathbf{X}}^T]$ can be utilized to stimulate the dynamic reactive spiking graph transformation layer.

Dynamic Reactive Spiking Graph Transformation

Recent brain's cognition researches (Deco, Cruzat, and Kringelbach 2019; Deco, Vidaurre, and Kringelbach 2021) observe that the reactive states of neurons may unevenly vary in the same tiny brain area. Therefore, guided by such spatial dynamic cognition in the brain when processing signals, we enforce our model aware of and realize such dynamic cognition. That is, we employ dynamic reactive neurons with unique optimizable thresholds inside our dynamic

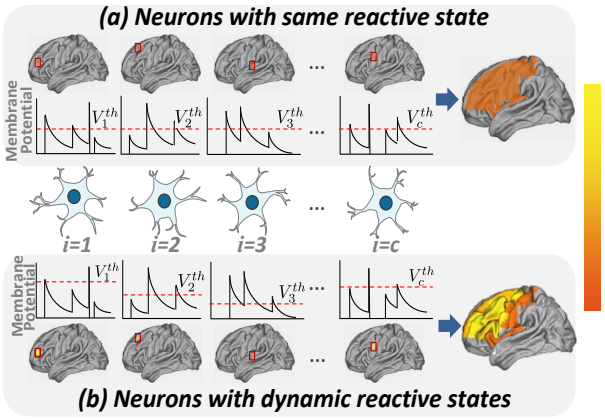

Figure 2: (a) The cognitive response corresponding to the existing same reactive graph spiking neural model (Zhu et al. 2022). (b) The cognitive response in the brain corresponding to our proposed dynamic reactive graph spiking neural model. Note that the darker the color in the brain, the more active it is in response to signals.

reactive spiking graph transformation (DRSGT) layer, thereby spontaneously exploring the spatial reactive dynamics between neurons. Concretely, stimulated by the spiking inputs $\hat{\mathbf{X}}^t$, the i-th post-neuron's output of our dynamic reactive spiking graph transformation model can be formulated as:

$$\boldsymbol{V}_i^t = \lambda \boldsymbol{V}_i^{t-1} + \sum_j W_{ij} \hat{\boldsymbol{X}}_j^t - V_i^{th} \boldsymbol{Z}_i^{t-1}, \tag{5}$$

$$Z_{i,k}^{t-1} = \begin{cases} 1, & \text{if } V_{i(rl),k}^{t-1} > 1 \\ 0, & \text{otherwise} \end{cases}, V_{i(rl),k}^{t-1} = \frac{V_{i,k}^{t-1}}{V_i^{th}}, \tag{6}$$

where $\hat{\boldsymbol{X}}_j^t$ is the n nodes' spiking input corresponding to the j-th pre-neuron at t-timestep, $\boldsymbol{V}_i^{t-1}$ and $\boldsymbol{Z}_i^{t-1}$ represent the n nodes' membrane potentials and spiking output vectors, respectively, of the i-th post-neuron at $(t-1)$-th timestep. $Z_{i,k}^{t-1}$ and $V_{i,k}^{t-1}$ are the k-th node's membrane potential and spiking output value of the i-th post-neuron at $(t-1)$-th timestep. $V_{i(rl),k}^{t-1}$ represent the k-th node's relative membrane potential compared with the threshold V_i^{th}. We denote $\mathbf{W} \in \mathbb{R}^{d \times c}$ and $\mathbf{V}^{th} \in \mathbb{R}^c$ as optimizable parameter matrices of this spiking transformation model, where W_{ij} is the weight value connecting j-th pre-neuron and i-th post-neuron in $\mathbf{W}$, and V_i^{th} represents the unique threshold value for i-th post-neuron in $\mathbf{V}^{th}$. The comparison illustration of neurons with the same and dynamic reactive states is provided in Figure 2, reflecting that our neuron-level dynamic method can capture more various reactive states like brains.

Learnable Graph Positional Spikes

Directly transforming the propagated-feature information by Equations 5 and 6 would still result in the underutilizing problem of long-range neighbor information. In other words, the feature propagation process with small steps cannot well aggregate the long-range-hop neighbor information. Meanwhile, directly increasing the step to aggregate the more far neighbors still suffers from the over-squashing problem (Alon and Yahav 2021); that is, the long-range neighbor information is compressed into a fixed-length feature vector. Such an underutilizing problem of long-range neighbors would lead to non-discriminative spiking graph outputs and poor model expressive ability. Therefore, to sufficiently explore the long-range neighbor information, inspired by Transformer (Vaswani et al. 2017), it is expected to integrate the positional graph information into our dynamic reactive spiking transformation layer. Due to the disordered property of graph data, there is no concept of absolute positional information in graph data. Therefore, the relative positional information, which can implicitly capture the neighbor structural similarity and discrepancy between nodes, is utilized to integrate into our spiking graph transformation process. Under the integration of the relative positional information, our model can capture the neighbor structural similarities and discrepancies between long-range nodes, thereby enhancing the model expression.

We design a learnable graph positional integration module to integrate positional information into the spiking graph outputs. We first initialize the positional information of the graph data by Laplacian eigenvectors (LSPE) (Dwivedi et al. 2020) or random walk (RWPE) (Dwivedi et al. 2022). Laplacian eigenvectors can capture distance-aware information:

$$\mathbf{PE}_k = [U_{k1}, U_{k2}, \cdots, U_{kd_{pos}}]^\top, \tag{7}$$

$$\Delta = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} = \mathbf{U}^\top \mathbf{\Lambda} \mathbf{U}, \tag{8}$$

where $\mathbf{U}$ and $\mathbf{\Lambda}$ are the Laplacian eigenvectors and eigenvalues, respectively. And the k-th node's d_{pos} smallest non-trivial eigenvectors are regarded as the k-th node's initial positional encoding, which is denoted as $\mathbf{PE}_k \in \mathbb{R}^{d_{pos}}$. U_{kr} $(r = 1, \cdots, d_{pos})$ represents the k-th node's r-th smallest non-trivial eigenvectors. d_{pos} reflects the dimension size of the spiking positional feature.

The other strategy to encode the graph positional information is the self random walk (Li et al. 2020):

$$\mathbf{PE}_k = [RW_{kk}, RW_{kk}^1, \cdots, RW_{kk}^{d_{pos}}]^\top. \tag{9}$$

The initial positional information of all nodes is also coded into the spiking form with T-duration: $[\hat{\mathbf{PE}}^1, \cdots, \hat{\mathbf{PE}}^T]$ through Poisson coding. Then, the initial spiking positional information is first integrated into our propagated-spiking feature inputs through concatenation:$[\hat{\mathbf{X}}^t; \hat{\mathbf{PE}}^t]$. And the dynamic reactive spiking graph transformation (Equation 5) is performed on such updated spiking position-integrated graph signals. In this way, the spiking positional information can be learned and fused adaptively with the propagated graph feature during the optimization of our spiking graph transformation layer, thereby guiding our model to efficiently explore the long-range neighbors.

Moreover, there is also another way to integrate spiking positional inputs into our spiking graph outputs. That is, an additional spiking neural network is constructed to

learn spiking positional information individually, whose basic neuron model is also dynamic reactive spiking one:

$$\boldsymbol{V}_{i_{pe}}^{t} = \lambda \boldsymbol{V}_{i_{pe}}^{t-1} + \sum_{j_{pe}} W'_{i_{pe}j_{pe}} \hat{\boldsymbol{PE}}_{j_{pe}}^{t} - V_{i_{pe}}^{th} \boldsymbol{Z}_{i_{pe}}^{t-1}, \quad (10)$$

$$Z_{i_{pe},k}^{t-1} = \begin{cases} 1, & \text{if} \quad V_{i_{pe}(rl),k}^{t-1} > 1 \\ 0, & \text{otherwise} \end{cases}, V_{i_{pe}(rl),k}^{t-1} = \frac{V_{i_{pe},k}^{t-1}}{V_{i_{pe}}^{th}}, \quad (11)$$

where i_{pe} and j_{pe} represent the i_{pe}-th post-neuron and j_{pe}-th pre-neuron in the dynamic reactive spiking neural network performed on spiking positional inputs. Then, such spiking positional information is fused to the learned graph embedding by addition, thereby enhancing the exploration ability on long-range neighbor information. Note that, our method uses RWPE initialization and the first integration way: concatenating the positional spikes with propagated graph spikes, then adaptively learning the position-integrated spiking outputs, by default in experiment section.

Overall Objective Function and Training

The T-duration output $[\boldsymbol{Z}_i^1, \boldsymbol{Z}_i^2, \cdots, \boldsymbol{Z}_i^T]$ of the i-th post neuron can be obtained after performing such dynamic reactive spiking graph neural network with learnable positional spikes. Then, we stack and normalize the T-duration output to get the predicted output logit $\mathbf{Y}^{pred}$: $\boldsymbol{Y}_i^{pred} = \sum_{t=1}^{T} \mathbf{Z}_i^t / \sum_{i=1}^{c} \sum_{t=1}^{T} \mathbf{Z}_i^t$, where $\boldsymbol{Y}_i^{pred}$ is the prediction vector of n nodes corresponding to the i-th class in $\mathbf{Y}^{pred}$.

The objective function is the prediction loss function $L = MSE(\mathbf{Y}^{pred}, \mathbf{Y})$, where $MSE(\cdot, \cdot)$ is the Mean Squared Error loss function. The gradient during backpropagation in the training process is $[\nabla \mathbf{W}, \nabla \mathbf{V}^{th}]$. Note that in the following content, we take the weight value W_{ij} between j-th pre-neuron and i-th post-neuron and the i-th post-neuron's firing threshold V_i^{th} as examples of gradient calculation.

The weight update is computed by $W_{ij} = W_{ij} - lr \nabla W_{ij}$, and ∇W_{ij} is computed by:

$$\begin{aligned} \nabla W_{ij} &= \frac{\partial L}{\partial W_{ij}} = \sum_k \sum_t \frac{\partial L}{\partial Z_{i,k}^t} \frac{\partial Z_{i,k}^t}{\partial V_{i(rl),k}^t} \frac{\partial V_{i(rl),k}^t}{\partial V_{i,k}^t} \frac{\partial V_{i,k}^t}{\partial W_{ij}} \\ &= \sum_k \sum_t \frac{\partial L}{\partial Z_{i,k}^t} \frac{\partial Z_{i,k}^t}{\partial V_{i(rl),k}^t} \frac{1}{V_i'^{th}} \hat{X}_{j,k}^t, \end{aligned} \quad (12)$$

where lr is the learning rate, $\hat{X}_{j,k}^t$ is the k-th node's spiking input corresponding to the j-th pre-neuron at t-timestep.

The threshold update is computed by $V_i^{th} = V_i^{th} - lr \nabla V_i^{th}$, and ∇V_i^{th} is computed by:

$$\begin{aligned} \nabla V_i^{th} &= \frac{\partial L}{\partial V_i^{th}} = \sum_k \sum_t \frac{\partial L}{\partial Z_{i,k}^t} \frac{\partial Z_{i,k}^t}{\partial V_{i(rl),k}^t} \frac{\partial V_{i(rl),k}^t}{\partial V_i'^{th}} \frac{\partial V_i'^{th}}{\partial V_i^{th}} \\ &= \sum_k \sum_t \frac{\partial L}{\partial Z_{i,k}^t} \frac{\partial Z_{i,k}^t}{\partial V_{i(rl),k}^t} \frac{(V_i'^{th} Z_{i,k}^{t-1} + V_{i,k}^t)(V_i'^{th} - 1)}{V_i'^{th}}. \end{aligned} \quad (13)$$

Algorithm 1: The training procedure of our proposed model

Input: $\mathbf{X}$, $\mathbf{A}$, $\mathbf{Y}_{train}$, and T
Output: Optimized $\boldsymbol{W}$, $\boldsymbol{V}^{th}$

```
Obtain propagated feature X̂ // Equation 1
Obtain initial positional feature PE // Equation 9
Random initialize W and V^th
while not converged do
  for t = 1 to T do
    Generate X̂^t and P̂E^t by Poisson coding
    Integrate P̂E^t into X̂^t through concatenation
    Reset and Chrge the DRSGT layer by position-integrated spiking inputs // Equation 5
    Fire spiking outputs // Equation 6
  end for
  Obtain final predictions Y^pred
  Update W and V^th // Equations 12 and 13
end while
```

To realize the backpropagation in the optimization of our proposed model, we replace the discontinuous gradient $\partial Z_{i,k}^t / \partial V_{i(rl),k}^t$ in Eqs. 12 and 13, from the pseudo-derivative (Wang, Cheng, and Lim 2022). In this way, we can update $\mathbf{W}$ and $\mathbf{V}^{th}$ during training to finally obtain a powerful model. The whole training procedure is provided in Algorithm 1.

Experiments

We have assessed our model learning ability on various datasets, obtaining competitive results. This section summarizes datasets, experimental setup, and results analysis.

Datasets

We use twelve graph datasets: Cora (McCallum et al. 2000), Citeseer (Sen et al. 2008), Pubmed (Namata et al. 2012), ogbn-arxiv (Hu et al. 2020), Amazon Photos, Amazon Computers, ACM (Fan et al. 2020), DBLP [1], Co-author C-S (Shchur et al. 2018), Co-author Physics (Shchur et al. 2018), flickr (Huang, Li, and Hu 2017), and blogcatalog (Huang, Li, and Hu 2017).

Experimental Setup

In the experiments, the splitting rules in three datasets: Cora, Citeseer, and Pubmed, are following (Kipf and Welling 2016), while the one of others is following the default one in datasets. The evaluation metrics for different datasets are testing the accuracy on the test set. For graph datasets, we utilize several artificial GNNs (GCN, SGC, DAGNN and so on) as the comparison methods, to show the superiority of our method in terms of model performance and energy consumption. And SpikingGCN (Zhu et al. 2022) is also regarded as the baselines.

[1] https://dblp.uni-trier.de/

	Method	Cora	Citeseer	Pubmed	ogbn-arxiv	Ama.Ph.	Ama.CS
Artificial ones	GCN (Kipf and Welling 2016)	81.35±1.03	69.93±1.21	78.09±0.62	71.82±0.30	86.56±0.28	79.31±0.47
	GAT (Veličković et al. 2017)	82.33±0.69	71.25±0.46	77.17±0.55	73.61±0.21	86.19±0.46	81.11±0.35
	SGC (Wu et al. 2019)	81.96±0.36	71.62±0.34	79.34±0.28	73.59±0.18	87.67±0.43	85.52±0.25
	FastGCN (Chen, Ma, and Xiao 2018)	80.36±0.98	70.02±0.76	76.99±0.58	**73.96±0.48**	85.86±0.25	85.47±0.40
	DAGNN (Liu, Gao, and Ji 2020)	84.01±0.57	72.06±0.52	79.62±0.35	73.05±0.49	89.19±0.43	86.36±0.38
	SDGNN (Zeng et al. 2021)	**85.25±0.30**	**74.35±0.47**	**80.37±0.48**	72.33±0.18	91.19±0.35	88.06±0.33
	GEM (Luo et al. 2023)	83.01±0.78	73.98±0.39	78.50±0.56	73.90±0.84	92.54±0.35	83.81±0.68
Spiking ones	SpikingGCN (Zhu et al. 2022)	77.72±0.65	70.58±0.54	77.14±0.53	62.95±0.22	87.01±0.62	88.35±0.34
	Ours	82.50±0.51	72.52±0.33	78.98±0.45	66.26±0.39	**92.80±0.55**	**89.41±0.36**
	Ours w/o LPSI	81.93±0.54	71.98±0.63	78.09±0.33	64.06±0.35	92.02±0.44	89.01±0.31
	Ours w/o DRM	80.62±0.37	71.23±0.32	78.02±0.35	64.33±0.29	91.82±0.36	87.93±0.37
	Ours r/w LLRM	77.79±0.49	70.39±0.40	77.52±0.46	63.98±0.25	89.52±0.70	88.21±0.46
	Method	ACM	DBLP	Co.CS	Co.Ph.	flickr	blogcatalog
Artificial ones	GCN (Kipf and Welling 2016)	93.30±0.72	82.56±0.33	90.73±0.46	95.05±0.37	57.46±0.48	72.51±0.46
	GAT (Veličković et al. 2017)	93.21±0.54	82.15±0.30	90.42±0.38	95.02±0.29	53.62±0.29	68.49±0.55
	SGC (Wu et al. 2019)	93.59±0.38	81.92±0.27	92.30±0.35	93.47±0.34	59.98±0.31	72.14±0.35
	FastGCN (Chen, Ma, and Xiao 2018)	93.90±0.46	82.08±0.31	92.21±0.27	95.23±0.56	52.31±0.41	66.86±0.39
	DAGNN (Liu, Gao, and Ji 2020)	94.06±0.33	82.76±0.47	92.56±0.26	93.48±0.25	**61.76±0.47**	73.92±0.37
	SDGNN (Zeng et al. 2021)	**94.49±0.23**	83.19±0.45	92.80±0.50	**96.02±0.19**	59.36±0.21	**74.06±0.32**
	GEM (Luo et al. 2023)	94.21±0.27	83.59±0.61	82.55±0.35	94.64±0.44	59.54±0.57	74.10±0.31
Spiking ones	SpikingGCN (Zhu et al. 2022)	91.06±0.45	83.04±0.33	91.40±0.45	93.99±0.37	57.69±0.29	70.68±0.50
	Ours	92.73±0.39	**83.95±0.30**	**92.85±0.38**	95.62±0.36	59.93±0.26	72.93±0.49
	Ours w/o LPSI	92.03±0.44	83.53±0.36	92.40±0.40	94.28±0.45	58.45±0.34	72.01±0.46
	Ours w/o DRM	91.65±0.30	83.41±0.38	92.16±0.39	94.36±0.36	58.81±0.39	71.98±0.65
	Ours r/w LLRM	91.13±0.43	83.20±0.19	91.07±0.60	94.06±0.31	58.04±0.27	70.79±0.42

Table 1: Classification performance (%) on various datasets.

Method	Cora	Citeseer	Pubmed	ogbn-arxiv	Ama.Ph	Ama.CS
unlearnable Lap.	69.62±0.43	69.86±0.24	68.47±0.60	61.36±0.53	75.12±0.40	77.42±0.36
unlearnable RW	70.06±0.37	69.75±0.30	70.50±0.53	60.99±0.48	80.01±0.40	80.66±0.33
learnable Lap._1	82.62±0.32	72.76±0.29	79.01±0.51	65.97±0.32	92.65±0.43	89.35±0.26
learnable RW_1	82.50±0.51	72.52±0.33	78.98±0.48	66.26±0.39	92.80±0.55	89.41±0.36
learnable Lap._2	82.44±0.49	72.61±0.32	78.69±0.46	66.01±0.29	92.76±0.37	89.65±0.39
learnable RW_2	82.53±0.36	72.92±0.37	78.83±0.49	66.41±0.40	93.14±0.40	89.86±0.35
Method	ACM	DBLP	Co.CS	Co.Ph.	flickr	blogcatalog
unlearnable Lap.	83.63±0.29	76.55±0.30	80.62±0.43	87.95±0.32	54.39±0.52	68.06±0.38
unlearnable RW	84.42±0.31	76.74±0.35	82.37±0.46	90.93±0.32	55.69±0.43	68.89±0.34
learnable Lap._1	92.34±0.37	83.64±0.20	92.60±0.35	95.67±0.37	59.91±0.33	72.63±0.45
learnable RW_1	92.73±0.39	83.95±0.30	92.85±0.38	95.62±0.36	59.93±0.26	72.93±0.49
learnable Lap._2	92.76±0.29	84.10±0.33	92.79±0.40	96.01±0.41	59.34±0.41	72.96±0.42
learnable RW_2	93.01±0.36	83.86±0.28	93.03±0.49	95.86±0.40	58.98±0.46	72.88±0.39

Table 2: Effect of positional initial and integrated ways on performance (%). 'Lap.' and 'RW' are positional initialization using Laplacian eigenvector and random walk. Subscripts 1 and 2 are different integrated ways mentioned, in turn, in section .

Results Analysis

In Table 1, our method is consistently comparable with the SNNs-belonging GNN-SpikingGCN (Zhu et al. 2022) on various datasets, and outperforms the traditional GNNs in most cases. Such a phenomenon indicates that our dynamic reactive Spiking GNN can generate more discriminative spiking predictions under our dynamic reactive spiking reactive mechanism and the sufficient exploration of the long-range neighbor information. Meanwhile, we evaluate the model's generalization capability by active learning, which discovers an acquisition function for successively selecting unlabeled data to optimize the model performance. We adopt three acquisition ways: σ-optimal acquisition function (SOPT) (Ma, Garnett, and Schneider 2013), standard predictive entropy one(PE) (Hernández-Lobato, Hoffman, and Ghahramani 2014), and random selection of samples; and the Area under the Learning Curve (ALC) is reported on GCN (Kipf and Welling 2016), SpikingGCN (Zhu et al. 2022), and our method in Table 3, which show our approach can gain better generalization capability than baselines.

Ablation Study. We evaluate the effectiveness of the designed neuron-level dynamic reactive mechanism (DRM) and learnable positional spiking integration (LPSI) in Table 1. Our neuron-level DRM can improve the model learning ability under our employed threshold optimization strat-

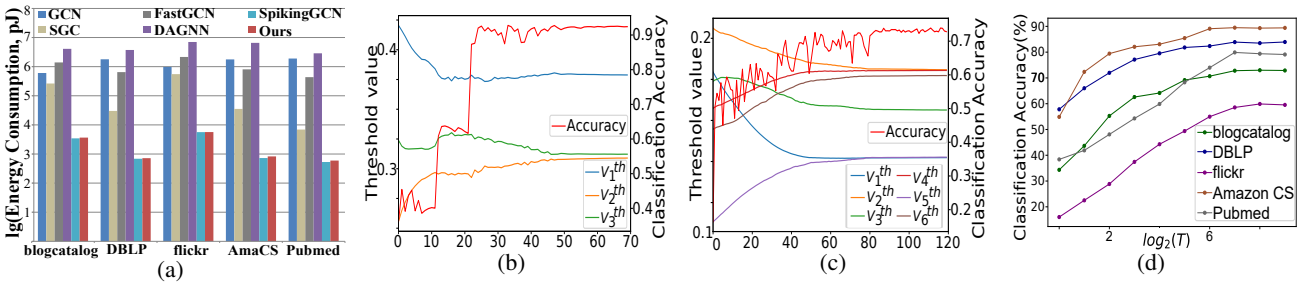

Figure 3: (a) Energy consumption comparisons. (b) The changing curve of each neuron's threshold during optimization on ACM. (c) The curve of each neuron's threshold during optimization on blogcatalog. (d) Parameter analysis on T.

Models	Cora	ACM
SOPT-GCN	71.01±0.32	85.37±0.64
SOPT-SpikingGCN	72.09±0.36	86.96±0.62
SOPT-Ours	**73.62±0.29**	**88.05±0.58**
PE-GCN	59.54±1.26	83.16±0.98
PE-SpikingGCN	62.64±1.32	84.97±1.15
PE-Ours	**64.07±1.35**	**85.94±0.89**
Random-GCN	56.99±2.26	82.43±1.76
Random-SpikingGCN	60.82±2.05	84.77±1.59
Random-Ours	**62.33±2.06**	**85.46±2.08**

Table 3: The area under ALC on several datasets.

egy, which reflects that such brain's dynamic reactive neuron mechanism is necessary for the spiking GNN model. In addition, the comparison between our method and the one removing LPSI can reflect that, our model can capture more long-range information with the integrated positional information in the dynamic spiking graph transformation process. Moreover, we also explore the model performance with different integrated strategies and positional initialization ways in Table 2. Note that 'unlearnable Lap.' and 'unlearnable R-W' in Table 2 represent the ones that first obtain the initial positional information $\mathbf{PE}$ with c-dimension, then directly fuse the spiking form $[\hat{\mathbf{PE}}^1, \cdots, \hat{\mathbf{PE}}^T]$ ($\hat{\mathbf{PE}}^t \in \mathbb{R}^{n\times c}, t = 1, \cdots, T$) into the spiking output by adding operation. Our model with learnable spiking positional signals can always achieve better performance than the unlearnable ones, which may be on account that the unlearnable positional signals cannot adaptively be integrated with our spiking graph signals to generate more discriminative outputs. Moreover, we compare our method with 'Ours r/w LLRM', which represents that the neuron-level reactive strategy in our model is replaced with the layer-level one.

Energy Consumption. Our model mainly includes a spiking graph feature transformation layer, and parameters of both our proposed dynamic reactive mechanism and learnable positional spikes are almost negligible for the entire model's parameter number. In this way, our energy consumption is in the same order as SpikingGCN, which can save energy 10 times at least with the traditional GNNs (Kipf and Welling 2016; Chen, Ma, and Xiao 2018; Wu et al. 2019; Liu, Gao, and Ji 2020). The comparisons of energy consumption with other methods are depicted in Figure 3(a). Therefore, although the capabilities of our model on some datasets in Table 1 have a little gap with existing ANNs-belonging GNNs, our method with relatively low energy consumption still has an advantage in practical applications.

Neuron Threshold Optimization. We have shown the changing curve of each neuron's threshold during the optimization process, as shown in Figures 3(b) and 3(c). These curves can reveal that during the optimization process, our method can spontaneously explore spatial-level dynamic reactive states between multiple neurons.

Hyperparameter Analysis. We also explore the influence of the hyper-parameter T on the graph classification effect on five datasets with different related fields, depicted in Figure 3(d). Figure 3(d) shows that the model's performance would be enhanced with T when T is small, which is on account that the spiking signal generated with less T may not sufficiently contain the information of the float propagated features. And the spiking signal generated with T can sufficiently capture the information of the float features when T reaches a certain value, so that the performance would no longer be affected by T in this case.

Conclusion

In this paper, we propose a dynamic reactive spiking graph neural network to achieve powerful learning ability with high bio-fidelity and energy efficiency. We adopt spiking neurons with different optimizable thresholds to spontaneously explore the reactive dynamics between neurons. Moreover, discriminative positional graph information is learned through our designed neurons, which is also integrated into spiking graph outputs to capture more long-range neighbor information and thus enhance model expression.

Acknowledgments

Our work was supported by Joint Fund of Ministry of Education of China (8091B022149), Key Research and Development Program of Shaanxi (2021ZDLGY01-03), National Natural Science Foundation of China (62132016, 62171343, 62071361 and 62201436), and Fundamental Research Funds for the Central Universities (ZDRC2102).

References

Alon, U.; and Yahav, E. 2021. On the bottleneck of graph neural networks and its practical implications. *Proc. Int. Conf. Learn. Represent.*

Anthony, L. F. W.; Kanding, B.; and Selvan, R. 2020. Carbontracker: Tracking and predicting the carbon footprint of training deep learning models. *arXiv preprint arXiv:2007.03051*.

Bellec, G.; Salaj, D.; Subramoney, A.; Legenstein, R.; and Maass, W. 2018. Long short-term memory and learning-to-learn in networks of spiking neurons. *Proc. Adv. Neural Inf. Process. Syst.*, 31.

Brette, R.; and Gerstner, W. 2005. Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *J. Neurophysiol.*, 94(5): 3637–3642.

Bruna, J.; Zaremba, W.; Szlam, A.; and LeCun, Y. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*.

Cao, Y.; Chen, Y.; and Khosla, D. 2015. Spiking deep convolutional neural networks for energy-efficient object recognition. *Int. J. Comput. Vis.*, 113: 54–66.

Chen, J.; Ma, T.; and Xiao, C. 2018. Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247*.

Chen, M.; Wei, Z.; Huang, Z.; Ding, B.; and Li, Y. 2020. Simple and Deep Graph Convolutional Networks. *arXiv preprint arXiv:2007.02133*.

Cheng, X.; Hao, Y.; Xu, J.; and Xu, B. 2020. LISNN: Improving spiking neural networks with lateral interactions for robust object recognition. In *Proc. Int. Jt. Conf. Artif. Intell.*, 1519–1525.

Cui, H.; Lu, Z.; Li, P.; and Yang, C. 2022. On positional and structural node features for graph neural networks on non-attributed graphs. In *Proc. Int. Conf. Inf. Knowl. Manag.*, 3898–3902.

Deco, G.; Cruzat, J.; and Kringelbach, M. L. 2019. Brain songs framework used for discovering the relevant timescale of the human brain. *Nat. Commun.*, 10(1): 583.

Deco, G.; Vidaurre, D.; and Kringelbach, M. L. 2021. Revisiting the global workspace orchestrating the hierarchical organization of the human brain. *Nat. Hum. Behav.*, 5(4): 497–511.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proc. Adv. Neural Inf. Process. Syst.*, 3844–3852.

Diehl, P. U.; Neil, D.; Binas, J.; Cook, M.; Liu, S.-C.; and Pfeiffer, M. 2015. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *Proc. Int. Jt. Conf. Neural Netw.*, 1–8. ieee.

Dwivedi, V. P.; Joshi, C. K.; Laurent, T.; Bengio, Y.; and Bresson, X. 2020. Benchmarking graph neural networks.

Dwivedi, V. P.; Luu, A. T.; Laurent, T.; Bengio, Y.; and Bresson, X. 2022. Graph neural networks with learnable structural and positional representations. *Proc. Int. Conf. Learn. Represent.*

Fan, S.; Wang, X.; Shi, C.; Lu, E.; Lin, K.; and Wang, B. 2020. One2multi graph autoencoder for multi-view graph clustering. In *Proc. Web Conf.*, 3070–3076.

Fang, W.; Yu, Z.; Chen, Y.; Masquelier, T.; Huang, T.; and Tian, Y. 2021. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *Proc. IEEE Int. Conf. Comput. Vis.*, 2661–2671.

Fourcaud-Trocmé, N.; Hansel, D.; Van Vreeswijk, C.; and Brunel, N. 2003. How spike generation mechanisms determine the neuronal response to fluctuating inputs. *J. Neurosci.*, 23(37): 11628–11640.

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

Gori, M.; Monfardini, G.; and Scarselli, F. 2005. A new model for learning in graph domains. In *Proc. Int. Jt. Conf. Neural Netw.*, volume 2, 729–734. IEEE.

Hernández-Lobato, J. M.; Hoffman, M. W.; and Ghahramani, Z. 2014. Predictive entropy search for efficient global optimization of black-box functions. *Proc. Adv. Neural Inf. Process. Syst.*, 27.

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural Comput.*, 9(8): 1735–1780.

Hu, W.; Fey, M.; Zitnik, M.; Dong, Y.; Ren, H.; Liu, B.; Catasta, M.; and Leskovec, J. 2020. Open graph benchmark: Datasets for machine learning on graphs. *Proc. Adv. Neural Inf. Process. Syst.*, 33: 22118–22133.

Huang, X.; Li, J.; and Hu, X. 2017. Label Informed Attributed Network Embedding. In de Rijke, M.; Shokouhi, M.; Tomkins, A.; and Zhang, M., eds., *Proc. Int. Conf. Web Search Data Min.*, 731–739. ACM.

Jin, Y.; Zhang, W.; and Li, P. 2018. Hybrid macro/micro level backpropagation for training deep spiking neural networks. *Proc. Adv. Neural Inf. Process. Syst.*, 31.

Kheradpisheh, S. R.; and Masquelier, T. 2020. Temporal backpropagation for spiking neural networks with one spike per neuron. *Int. J. Neural Syst.*, 30(06): 2050027.

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

Kreuzer, D.; Beaini, D.; Hamilton, W.; Létourneau, V.; and Tossou, P. 2021. Rethinking graph transformers with spectral attention. *Proc. Adv. Neural Inf. Process. Syst.*, 34.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11): 2278–2324.

Lee, J. H.; Delbruck, T.; and Pfeiffer, M. 2016. Training deep spiking neural networks using backpropagation. *Front. Neurosci.*, 10: 508.

Li, P.; Wang, Y.; Wang, H.; and Leskovec, J. 2020. Distance encoding: Design provably more powerful neural networks for graph representation learning. *Proc. Adv. Neural Inf. Process. Syst.*, 33: 4465–4478.

Liu, M.; Gao, H.; and Ji, S. 2020. Towards deeper graph neural networks. In *Proc. Int. Conf. Knowl. Discov. Data Min.*, 338–348.

Luo, Y.; Luo, G.; Qin, K.; and Chen, A. 2023. Graph Entropy Minimization for Semi-supervised Node Classification. *arXiv preprint arXiv:2305.19502*.

Ma, Y.; Garnett, R.; and Schneider, J. 2013. σ-optimality for active learning on gaussian random fields. *Proc. Adv. Neural Inf. Process. Syst.*, 26.

McCallum, A. K.; Nigam, K.; Rennie, J.; and Seymore, K. 2000. Automating the construction of internet portals with machine learning. *Inf. Retr.*, 3: 127–163.

Namata, G.; London, B.; Getoor, L.; Huang, B.; and Edu, U. 2012. Query-driven active surveying for collective classification. In *Int. Workshop on Min. Learn. Graphs*, volume 8.

Neftci, E. O.; Mostafa, H.; and Zenke, F. 2019. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Process. Mag.*, 36(6): 51–63.

Rathi, N.; and Roy, K. 2020. Diet-snn: Direct input encoding with leakage and threshold optimization in deep spiking neural networks. *arXiv preprint arXiv:2008.03658*.

Ratliff, F.; Hartline, H. K.; and Lange, D. 1974. The dynamics of lateral inhibition in the compound eye of Limulus. I. *Studies on Excitation and Inhibition in the Retina: A Collection of Papers from the Laboratories of H. Keffer Hartline*, 463.

Rueckauer, B.; Lungu, I.-A.; Hu, Y.; Pfeiffer, M.; and Liu, S.-C. 2017. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Front. Neurosci.*, 11: 682.

Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; and Monfardini, G. 2008. Computational capabilities of graph neural networks. *IEEE Trans. Neural Netw.*, 20(1): 81–102.

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; and Eliassi-Rad, T. 2008. Collective classification in network data. *AI Mag.*, 29(3): 93–93.

Shchur, O.; Mumme, M.; Bojchevski, A.; and Günnemann, S. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Proc. Adv. Neural Inf. Process. Syst.*, 30.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Wang, B.; and Jiang, B. 2022. Spiking GATs: Learning Graph Attentions via Spiking Neural Network. *arXiv preprint arXiv:2209.13539*.

Wang, S.; Cheng, T. H.; and Lim, M.-H. 2022. LTMD: Learning Improvement of Spiking Neural Networks with Learnable Thresholding Neurons and Moderate Dropout. *Proc. Adv. Neural Inf. Process. Syst.*, 35: 28350–28362.

Wu, F.; Souza, A.; Zhang, T.; Fifty, C.; Yu, T.; and Weinberger, K. 2019. Simplifying graph convolutional networks. In *Proc. Int. Conf. Mach. Learn.*, 6861–6871.

Wu, Y.; Deng, L.; Li, G.; Zhu, J.; and Shi, L. 2018. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Front. Neurosci.*, 12: 331.

Xu, M.; Wu, Y.; Deng, L.; Liu, F.; Li, G.; and Pei, J. 2021. Exploiting spiking dynamics with spatial-temporal feature normalization in graph learning. *arXiv preprint arXiv:2107.06865*.

Yang, X.; Deng, C.; Liu, T.; and Tao, D. 2020a. Heterogeneous graph attention network for unsupervised multiple-target domain adaptation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(4): 1992–2003.

Yang, Y.; Feng, Z.; Song, M.; and Wang, X. 2020b. Factorizable graph convolutional networks. *Proc. Adv. Neural Inf. Process. Syst.*

Zeng, H.; Zhang, M.; Xia, Y.; Srivastava, A.; Malevich, A.; Kannan, R.; Prasanna, V.; Jin, L.; and Chen, R. 2021. Decoupling the depth and scope of graph neural networks. *Proc. Adv. Neural Inf. Process. Syst.*, 34: 19665–19679.

Zhu, Z.; Peng, J.; Li, J.; Chen, L.; Yu, Q.; and Luo, S. 2022. Spiking graph convolutional networks. *arXiv preprint arXiv:2205.02767*.