Exploiting Spiking Dynamics with Spatial-temporal Feature Normalization in Graph Learning

Mingkun Xu[*] , **Yujie Wu**[*] , **Lei Deng** , **Faqiang Liu** , **Guoqi Li** and **Jing Pei**[†]

Center for Brain-Inspired Computing Research (CBICR), Beijing Innovation Center
for Future Chip, Department of Precision Instrument, Tsinghua University

{xmk18,wu-yj16,lfq18}@mails.tsinghua.edu.cn, {leideng,liguoqi,peij}@mail.tsinghua.edu.cn

Abstract

Biological spiking neurons with intrinsic dynamics underlie the powerful representation and learning capabilities of the brain for processing multimodal information in complex environments. Despite recent tremendous progress in spiking neural networks (SNNs) for handling Euclidean-space tasks, it still remains challenging to exploit SNNs in processing non-Euclidean-space data represented by graph data, mainly due to the lack of effective modeling framework and useful training techniques. Here we present a general spike-based modeling framework that enables the direct training of SNNs for graph learning. Through spatial-temporal unfolding for spiking data flows of node features, we incorporate graph convolution filters into spiking dynamics and formalize a synergistic learning paradigm. Considering the unique features of spike representation and spiking dynamics, we propose a spatial-temporal feature normalization (STFN) technique suitable for SNN to accelerate convergence. We instantiate our methods into two spiking graph models, including graph convolution SNNs and graph attention SNNs, and validate their performance on three node-classification benchmarks, including Cora, Citeseer, and Pubmed. Our model can achieve comparable performance with the state-of-the-art graph neural network (GNN) models with much lower computation costs, demonstrating great benefits for the execution on neuromorphic hardware and prompting neuromorphic applications in graphical scenarios.

1 Introduction

Spiking neural network (SNN) is a typical brain-inspired network, which inherits the biological spatio-temporal dynamics mechanisms and rich spiking coding schemes, demonstrating great benefits for mimicking neuroscience-inspired models and performing efficient computation in neuromorphic hardware. Recently SNNs have been successfully applied in a variety of applications, such as image classification,

voice recognition, object tracking, neuromorphic perception, etc [Wu *et al.*, 2018; Silva *et al.*, 2017; Cao *et al.*, 2015; Zhu *et al.*, 2020; Haessig *et al.*, 2019]. However, most of published works have mainly focused on processing unstructured information represented in the Euclidean space, such as image, language data, etc [Zhou *et al.*, 2018]. On the other hand, biological spiking neurons with rich coding schemes and spatio-temporal dynamics underlie the foundation of graphic cognitive map in the hippocampal-entorhinal system. It facilitates the organization and assimilation of internal association attributes in structured spatial representations for performing relational memory and advanced cognition [Whittington *et al.*, 2020]. Hence, it is highly expected to explore an effective algorithm for training SNNs involved with topology relations and structural knowledge.

Recently, there exists some works of combining SNNs with graphic scenarios, whereas these studies mainly focused on applying graph theory to analyze the features of spiking neuron and network topology [Piekniewski, 2007; Cancan, 2019; Jovanović and Rotter, 2016], or using the features of spiking neurons to solve simple graph-related problems, such as the shortest path problems, clustering problems, minimal spanning tree problems [Sala and Cios, 1999; Hamilton *et al.*, 2020]. More recently, the work [Gu *et al.*, 2020] introduced a graph convolution to pre-process tactile data and trained an SNN classifier. Although they achieved high performance on the classification of sensor data, this method is difficult to accommodate general graph operations and transfer to other scenarios. On the other hand, plenty of GNN models have been developed, such as Graph Convolution Network (GCN) and Graph Attention Network (GAT), and obtained remarkable progress in solving graph-related tasks [Kipf and Welling, 2019; Veličković *et al.*, 2017]. While few works considered the relationship between graph theory and neural dynamics, and most of them suffered from a huge computational overhead when applied in scenarios with large-scale data, potentially limiting practical applications.

To explore the feasibility and advantages of spiking dynamics for handling graph-structured data, here we report a graph SNN framework capable of processing non-Euclidean data with general operations by introducing versatile aggregation methods. To develop such SNN models, two main challenges need to overcome. The first is to reconcile multiple graph convolution operations and spiking dynamics. To

---

[*]Contribute equally, [†]Corresponding author

this end, we firstly unfold the data flows of binary node features along the temporal dimension and spatial dimension, and propose a general spiking message passing method incorporating graph filter and spiking dynamics in an iterative manner. The second is to ensure the training convergence and performance for graph SNNs. Because of the complex spiking dynamics and diverse graph operations, directly training SNNs on graph is still rarely researched and challenging. Although various normalization techniques have demonstrated their powerful effectiveness for the network convergence [Ioffe and Szegedy, 2015; Ba *et al.*, 2016], it is difficult to apply these techniques directly to graph SNNs especially owning to temporal neuronal dynamics and binary communication mechanism [Zheng *et al.*, 2020]. Therefore, it is necessary to develop a new normalization method suitable for spiking dynamics and graph scenarios. To address this problem, we propose a spatial-temporal feature normalization algorithm (STFN) by normalizing instant membrane potentials corresponding to each node in feature dimension and temporal dimension. In this manner, SNNs can gain a powerful capacity for abstracting hidden features from aggregated signals in a graph. Overall, our contributions are summarized as follows:

- We build a general spike-based modeling framework, referred to as Graph SNNs, to support spike propagation and feature affine transformation by reconciling the graph convolution operation and spiking communication mechanism. The framework is transferable and can accommodate most of graph propagation operations. To our best knowledge, this is the first work to build a general spike-based framework for processing regular graph tasks in gradient-descent paradigm.

- We propose a spatial-temporal feature normalization algorithm (STFN), which can incorporate the temporal neuronal dynamics and coordinate the membrane potential representation with threshold, demonstrating a significant improvement on convergence and performance.

- We instantiate the proposed framework into several models (GC-SNN and GA-SNN) and demonstrate their performance by multiple experiments on the semi-supervised node classification tasks. We further evaluate the computation costs and demonstrate high-efficiency benefits which may bring new opportunity to unlock SNNs potentials and facilitate graph-structured applications for neuromorphic hardware.

2 Graph Spiking Neural Networks

We introduce our Graph SNN framework mainly from following four aspects: (1) an approach to accommodate graph convolution operation into spiking computing paradigm; (2) an iterative spiking message passing model and the overall training framework containing two phases; (3) the spatial-temporal feature normalization for facilitate convergence; (4) instantiations of specific spiking graph model.

2.1 Spiking Graph Convolution

Given an attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\mathcal{V}$ is the set of nodes, and $\mathcal{E}$ is the set of edges. Generally, the graph attributes can be described by an adjacent matrix $A \in \mathbb{R}^{N \times N}$ and a node feature matrix $X \in \mathbb{R}^{N \times C} = [x_1; x_2; ...; x_N]$, where each row of X represents the feature vector of each node x_i and the adjacent matrix A satisfies $A_{ii} = 0$. Assume the single node signal is represented as a scalar vector $x \in \mathbb{R}^C$, the spectral convolution is defined as the multiplication of a filter $\mathcal{F}_\theta = diag(\theta)$ and node signal x as below:

$$\mathcal{F}_\theta \star x = U \mathcal{F}_\theta U^\top x, \tag{1}$$

where $\mathcal{F}_\theta$ is parameterized by $\theta \in \mathbb{R}^C$, U is the eigenvector matrix of Laplacian matrix $L = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} = U \Lambda U^\top$. $U^\top x$ can be regarded as the graph Fourier transform of x, and the diagonal matrix of eigenvalues Λ can be filtered by the function $\mathcal{F}_\theta(\Lambda)$. Due to the expensive computation overhead, some work proposed to use approximation and stacked multiple nonlinear layers to circumvent it, which has been applied successfully recently [Kipf and Welling, 2019].

To unify the information flows in spiking dynamics, we encode initial node signals X into binary components $\{\widetilde{X}_0, \widetilde{X}_1, ..., \widetilde{X}_t, ..., \widetilde{X}_{T-1}\}$, where T signifies the length of time window. The encoding method can be probabilistic, such as following a Bernoulli distribution or Poisson distribution, and can also be deterministic, such as quantitative approximation or using an encoding layer to generate spikes [Wu *et al.*, 2019]. In this manner, we can transform an attributed graph from its initial state into a spiking expression. We use $\widetilde{H}_t^n$ to represent the spiking node embedding in the n_{th} layer at time t, and $\widetilde{H}_t^0 = \widetilde{X}_t$ (tilde denotes binary variable in spike form). Then the layer-wise spiking graph convolution in spatial-temporal domain can be defined as:

$$\widetilde{H}_t^n = \Phi(\mathcal{G}_c(A, \widetilde{H}_t^{n-1}) W^n, \widetilde{H}_{t-1}^n), \tag{2}$$

where $\mathcal{G}_c(A, \widetilde{H})$ denotes the spiking feature propagation on the topology connection structure, which can be implemented via various propagation methods, such as Chebyshev filter, 1st model, etc. $\Phi(\cdot)$ is the non-linear dynamic process related to historical states $\widetilde{H}_{t-1}^n$. $W^n \in \mathbb{R}^{C^{n-1} \times C^n}$ is the layer-wise trainable weight parameter matrix, where C^n denotes the output dimension of spiking feature in the n_{th} layer and $C^0 = C$. We use the formula (2) to represent one spiking graph convolution layer. And multi-layer-stacked structure can form a comprehensive spiking dynamics system.

2.2 Iterative Spiking Message Passing

We adopt the leaky integrate-and-fire (LIF) model as our basic neuron unit, which is computationally tractable and most commonly used, meanwhile maintaining biological fidelity to a certain degree. Its neuronal dynamics can be governed by:

$$\tau \frac{dV(t)}{dt} = -(V(t) - V_{reset}) + I(t), \tag{3}$$

where $V(t)$ denotes the neuronal membrane potential at time t. τ is a time constant and $I(t)$ signifies the pre-synaptic input, which is obtained by the combined action of synaptic weights and pre-neuronal activities or external injecting stimulus. When $V(t)$ exceeds a certain threshold V_{th}, the neuron emits a spike and resets its membrane potential to V_{reset}, then starts to accumulate $V(t)$ again in subsequent time steps.
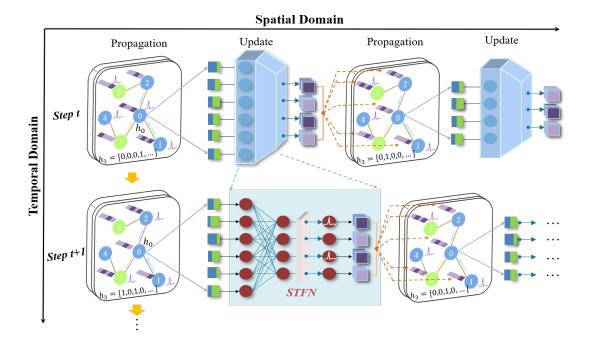
Figure 1: The Graph SNN framework can support spiking message propagation and feature affine transformation, reconciling the graph convolution operation and spiking communication mechanism in a unified paradigm. The proposed STFN normalizes the membrane potentials along both spatial and temporal dimension, which coordinates the data distribution with threshold but also facilitates the network convergence.

The spiking message passing process includes information propagation step and update step, both of which run for T time steps. Let $\widetilde{H}_t = [\widetilde{h}_t^0; \widetilde{h}_t^1; ...; \widetilde{h}_t^{N-1}] \in \mathbb{R}^{N \times C}$, for the node v, we provide a universal formulation for its spiking message passing as:

$$\widetilde{h}_t^v = U(\sum_{u \in N(v)} P(\widetilde{h}_t^v, \widetilde{h}_t^u, e_{vu}), \widetilde{h}_{t-1}^v)), \quad (4)$$

where $P(\cdot)$ represents the spiking message aggregation for neighbouring nodes, which can be implemented by different graph convolution operation $\mathcal{G}_c(\cdot)$. $U(\cdot)$ represents the state update via a non-linear dynamic system. $N(v)$ denotes all the neighbors of node v in graph $\mathcal{G}$, and e_{vu} denotes the static edge connection between v and u, which can be extended to directed multigraph formalism trivially. Equation (2) is a specific implementation of the above universal expression.

To accommodate the LIF model into the above paradigm, we use Euler method to transform the first-order differential equation of Eq.(3) into an iterative expression. We use a decay factor κ to represent the item $(1 - \frac{dt}{\tau})$ and expand the pre-synaptic input I as $\sum_j W^j \mathcal{G}_c(A, \widetilde{H}_{t+1}^j)$. Note that here we use graph convolution to realize propagation step $P(\cdot)$, then we obtain the following formulation via incorporating the scaling effect of $\frac{dt}{\tau}$ into weight item:

$$V_{t+1} = \kappa V_t + \sum_j W^j \mathcal{G}_c(A, \widetilde{H}_{t+1}^j). \quad (5)$$

$\mathcal{G}_c(A, \widetilde{H}_{t+1}^j)$ denotes the corresponding pre-synaptic aggregated feature and the superscript j denotes pre-synapse index.

By incorporating firing-and-resetting mechanism and assuming $V_{reset} = 0$, the update equation can be formulated as:

$$V_{t+1}^{n,i} = \kappa V_t^{n,i}(1 - \widetilde{H}_t^{n,i}) + \sum_j^{l(n)} W^{n,ij} \mathcal{G}_c(A, \widetilde{H}_{t+1}^{n-1,j}), \quad (6)$$

$$\widetilde{H}_{t+1}^{n,i} = g(V_{t+1}^{n,i} - V_{th}), \quad (7)$$

where n denotes the n_{th} layer and $l(n)$ denotes its corresponding neuron number. W^{ij} represents synaptic weight from j_{th} neuron in pre-layer to i_{th} in post-layer. $g(\cdot)$ is Heaviside step function. In this manner, we convert the implicit differential function into an explicit iterative version, which can describe the message propagation and update process in formula (4) and Figure 1.

Additionally, we find that the computation order of affine transformation and graph convolution can be exchanged when the graph convolution operation is linear (e.g. $\mathcal{G}_c(A, H) = D^{-\frac{1}{2}} A D^{-\frac{1}{2}} H$). Considering that W is quite dense, $\widetilde{H}$ is very sparse and binary, the prioritizing calculation for $\widetilde{H}$ and W will reduce computation overhead by turning multiplication into addition. In this situation, the process (6) can be re-formulated as:

$$V_{t+1}^{n,i} = \kappa V_t^{n,i}(1 - \widetilde{H}_t^{n,i}) + \mathcal{G}_c(A, \sum_j^{l(n)} W^{n,ij} \widetilde{H}_{t+1}^{n-1,j}). \quad (8)$$

In this manner, we provide a universal spiking passing framework in iterative manner. By specializing $\mathcal{G}_c(\cdot)$, most of proposed graph convolution operations [Kipf and Welling, 2019;

Gilmer *et al.*, 2017; Hamilton *et al.*, 2017] can be incorporated into this compatible model. And this framework is transferable to other graph scenarios.

2.3 Spatial-temporal Feature Normalization

Due to the additional temporal dynamics and event-driven spiking binary representation, traditional normalization techniques cannot be applied to the SNN framework directly. Moreover, on the other hand, the convergence and performance cannot be guaranteed by directly training SNNs on graph tasks. Therefore, it motivates us to propose a spatial-temporal feature normalization (STFN) algorithm specialized for the spiking dynamics on graph scenarios.

Considering the feature map calculation step, let $S_t \in \mathbb{R}^{N \times C}$ represent the instant outputting membrane potential of all neurons in a layer at time step t ($\sum_j^{l(n)} W^{n,ij} \mathcal{G}_c(A, \widetilde{H}_t^{n-1,j})$ in formula (6) or $\sum_j^{l(n)} W^{n,ij} \widetilde{H}_t^{n-1,j}$ in formula (8)). In STFN, the pre-synaptic inputs will be normalized along the feature dimension C independently for each node. Considering that the temporal effects play a significant role in operating node features transformation in topology space, we simultaneously normalize along the temporal dimension at consecutive time steps. Let $S_t^{k,v}$ represent the k_{th} element in feature vector of a node v. $S_t^{k,v}$ will be normalized by

$$\hat{S}_t^{k,v} = \frac{\rho V_{th}(S_t^{k,v} - E[S^v])}{\sqrt{Var[S^v] + \epsilon}},$$
$$Y_t^{k,v} = \lambda^{k,v} \hat{S}_t^{k,v} + \gamma^{k,v}, \quad (9)$$

where ρ is a hyper-parameter optimized in the training process and ϵ is a tiny constant. $\lambda^{k,v}$ and $\gamma^{k,v}$ are two trainable parameter vectors for node v. $E[S^v]$ and $Var[S^v]$ denote the mean and variance of node v statistically computed over feature dimension and temporal dimension, respectively. Figure 2 depicts the computation process of $E[S^v]$ and $Var[S^v]$, which are defined as follows:

$$E[S^v] = \frac{1}{CT} \sum_{t=0}^{T-1} \sum_{k=0}^{C-1} S_t^{k,v},$$
$$Var[S^v] = \frac{1}{CT} \sum_{t=0}^{T-1} \sum_{k=0}^{C-1} (S_t^{k,v} - E[S^v])^2. \quad (10)$$

We follow the above schema and find the normalization can facilitate convergence, improve performance in training.

2.4 Graph SNNs Framework

Our framework includes two phases: spiking message passing and readout, wherein the spiking message passing phase is described in Section 2.2. After the T times of iteration for the inference process are finished, the output spiking signals need to be decoded and transformed into high-level representations for downstream tasks. Under the rate coding condition, the readout phase computes the decoded feature vector for the whole graph using readout function R according to

$$\hat{y}^v = R(\{\frac{1}{T} \sum_{t=0}^{T-1} \widetilde{h}_t^v | v \in \mathcal{G}\}). \quad (11)$$
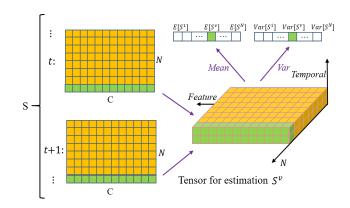

Figure 2: The schematic diagram of spatial-temporal feature normalization (STFN), where the high-dimensional pre-synapse inputs will be normalized along the feature dimension and temporal dimension.

Here, the readout function operates on the set of node states and should be invariant to permutations of the node states in order that Graph SNNs can be invariant to graph isomorphism. Besides, the readout function can be implemented via multiple approaches, such as using a differentiable function or neural network that maps to an output.

As shown in Figure 1, the first phase of our framework contains two unfolding domains, i.e., spatial domain and temporal domain. In the temporal domain, we utilize the encoded binary node spikes as the current node feature at time step t. A sequence of binary information will be processed successively, where the historical information for each neuron plays a key role in spiking dynamics and the decoding for output. In the spatial domain, the spiking features of a batch of nodes will be firstly aggregated via connected edges from one-hop neighbors, where the propagation operation is compatible with multiple proposed methods. Then the aggregated features will be fed into the spiking network module. In each network module, we conduct STFN to normalize the instant outputting membrane potential along the feature and temporal dimension, which ensures the node features of all dimensions follow the distribution $N(0, (\rho V_{th})^2)$. Each layer possesses the spatial-temporal dynamics for feature abstracting and mapping, and the stacked multi-layer structure will empower SNNs stronger representation capability. The inference procedure is provided in Supplementary Materials.

We take the semi-supervised learning task as example for illustration. With a small subset of node labels available, we evaluate the cross-entropy error over all labeled examples:

$$\mathcal{L} = - \sum_{l \in \mathcal{Y}_L} \sum_{r=0}^{C^L-1} Y_{lr} ln\sigma(\hat{y}_{lr}), \quad (12)$$

where $\sigma(\cdot)$ denotes softmax function, $\mathcal{Y}_L$ represents the set of node indices possessing labels. Y_{lr} denotes the real labels corresponding to the r_{th} dimension of C^L classes. In order to train the Graph SNN models effectively via gradient descent paradigm, we adopt the gradient substitution method in backward path [Wu *et al.*, 2018] and take the rectangular function to approximate the derivative of spike activity.

2.5 Model Instantiating: GC-SNN and GA-SNN

To demonstrate the effectiveness of our framework and normalization technique, we instantiate the framework into specific models by introducing specifically designed propagation methods most used in GNNs: graph convolution aggregator and graph attention mechanism [Kipf and Welling, 2019; Veličković *et al.*, 2017]. In this case, our Graph convolution Spiking Neural Network (GC-SNN) can be formulated as:

$$\widetilde{h}_t^{n,i} = \Phi\Big(\sum_{j \in \mathcal{N}(i)} \frac{1}{c_{ij}} \widetilde{h}_t^{n-1,j} W^n + b^n, \widetilde{h}_{t-1}^n\Big), \qquad (13)$$

where $\mathcal{N}(i)$ is the set of neighbors of node i, c_{ij} is the product of square root of node degrees (i.e. $c_{ij} = \sqrt{|\mathcal{N}(i)|}\sqrt{|\mathcal{N}(j)|}$). b^n is a trainable bias parameter. Similarly, our Graph Attention Spiking Neural Network (GA-SNN) can be expressed as:

$$\widetilde{h}_t^n = \Phi\Big(\alpha_{ij}^n \sum_{j \in \mathcal{N}(i)} \widetilde{h}_t^{n-1,j} W^n, \widetilde{h}_{t-1}^n\Big),$$

$$\alpha_{ij}^n = \frac{exp(f_l(a^{nT}(\widetilde{h}_t^{n-1,i} W^n || \widetilde{h}_t^{n-1,j} W^n)))}{\sum_{k \in \mathcal{N}(i)} exp(f_l(a^{nT}(\widetilde{h}_t^{n-1,i} W^n || \widetilde{h}_t^{n-1,k} W^n)))}, \qquad (14)$$

where $f_l(\cdot)$ denotes $LeakyReLu(\cdot)$ function. a^n is a learnable weight vector and $||$ denotes the concatenation operation. The core idea is to conduct attention mechanism on graph convolution and learn different importance degree α for each node. We exhibit the results of the above models for illustration in the experiment section.

3 Experiments

In this section, We firstly investigate the capability of our models over semi-supervised node classification on three citation datasets to examine their performance. Then we demonstrate the effectiveness of the proposed STFN and provide some analysis visualization for the model efficiency. We provide the experimental configuration details in Supplementary Materials.

3.1 Datasets and Pre-processing

We use three standard citation network benchmark datasets—Cora, Pubmed, and Citeseer, where nodes represent paper documents and edges are (undirected) citation links. We summarize the dataset statistics in Supplementary Materials. Note that from bag-of-words model, the features can be represented as binary vectors, which is equipped with similar attributes with spike representation. Here, we take the binary representation as spike vector, and re-organized them as an assembling sequence set w.r.t timing, where each component vector is considered equal at different time step for simplicity. In this manner, the original signals can be processed and treated as spiking signals.

3.2 Performance and Analysis

Basic Performance. For fairness, we take the same settings for GNN models (GCN, GAT) and our SNN models (GC-SNN, GA-SNN), and also keep the settings of each dataset

Method	Cora	Pubmed	Citeseer
DeepWalk	67.2	65.3	43.2
ICA	75.1	73.9	69.1
Planetoid	75.7	77.2	64.7
Chebyshev	81.2	74.4	69.8
GCN	81.5	79.0	70.3
GCN*	81.9±1.1	79.4±0.4	70.4±1.1
GAT*	82.3±0.6	78.4±0.5	71.1±0.2
GC-SNN(Ours)	**80.7**±0.6	77.9±0.5	**69.9**±0.9
GA-SNN(Ours)	79.7±0.6	**78.0**±0.4	69.1±0.5

Table 1: Performance comparison on benchmark datasets [Perozzi *et al.*, 2014; Getoor, 2005; Yang *et al.*, 2016; Defferrard *et al.*, 2016]. ∗ denotes the results in our implementation and ± denotes the standard deviation calculated from 10 runs.

Operations($\times 10^6$)	Cora	Pubmed	Citeseer
GNN(2 layers)	2.78	7.82	3.02
SNN(2 layers)	0.22	0.34	0.26
GNN(3 layers)	143.24	1018.66	175.84
SNN(3 layers)	3.63	24.36	3.28
Compre. ratio(2 layers) *	12.62×	23.00×	11.62×
Compre. ratio(3 layers) *	39.46×	41.82×	53.61×

Table 2: Operation comparison on benchmark datasets. ∗ denotes the compression ratio (GNN Opts. / SNN Opts.) in feature transformation process.

same. We report our results for 10 trials in Table 1. The results suggest that even if using binary spiking communication, our SNN models can achieve comparable performance with the SOTA results with a minor gap. It proves the feasibility and powerful capability of spiking mechanism and spatial-temporal dynamics, which can bind diversiform features from different nodes and work well on graph scenarios with few labels.

The Effects of STFN. We further visualize the data distribution in Figure 3(a), which indicates that the STFN can push the feature distribution to follow a larger standard deviation controlled by the threshold. In this case, the normalized membrane potential state can coordinate distribution difference and spike-triggered threshold. We find that this method contributes to the stability of performance via multiple experiments. Additionally, we plot the comparison curves for test accuracy and loss during the training process(Figure 3(b)(c)) on Citeseer. It also indicates that STFN can accelerate convergence, alleviate over-fitting, and significantly improve the performance on semi-supervised learning tasks.

The Effects of Time Window. We conduct multiple experiments with different time window lengths. As depicted in Figure 4(a), a longer time window length will benefit for the accuracies progressively, while the dependency on time window length is not prominent. Evidently, our model can still achieve comparable results even with a small time window and training cost.
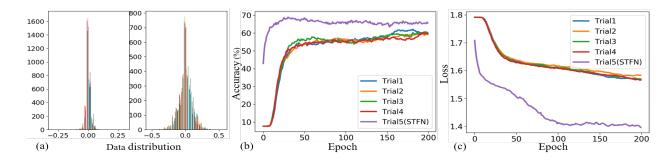
Figure 3: (a) The pre-activated membrane potential distribution before STFN (Left) and after STFN (Right). (b) Test accuracy comparison during training on Citeseer between Graph SNN models with STFN and without STFN. (c) Test loss comparison during training on Citeseer between Graph SNN models with STFN and without STFN.
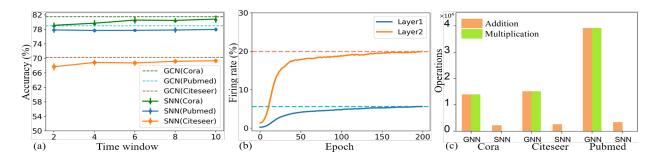

Figure 4: (a) Accuracy variation with respect to time window length (error bar denotes standard deviation from 10 runs). (b) Averaged firing rate variation during training process. (c) Operation cost comparison of feature transformation between GNNs and SNNs.

3.3 Efficiency

As presented in Figure $4(b)$, because we use the second layer for decoding and determining, the neuron activity density is much higher than that in the first layer. In the overall training process, the highest averaged firing rate is no more than 20%, and the overwhelming majority of neurons fire less than 10% (see Appendix), verifying the prominent sparsity of spike features. More details are provided in Supplementary Materials.

We compare the affine transformation costs of GNN network and SNN network with 2-layer structure on three datasets, respectively. As shown in Figure $4(c)$, our SNN models have no addition operation and achieve far fewer multiplication operations than GNN models (compression ratio $11.62\times \sim 23.00\times$). In Table 2, the advantages become more prominent with deeper and larger network structure where the compression ratio become $39.46\times \sim 53.61\times$. Fundamentally, the spike communication and even-driven computation underlie the efficiency advantage of SNNs. On the one hand, the binary spiking feature can transform the matrix manipulation with dense weights from multiplication into addition, where the energy consumption for addition is less than multiplication on neuromorphic hardware. On the other hand, the event-driven characteristics enable sparse firing and representation for node features, reducing the operation costs by a large margin. Besides, since the spike-based communication and local computation can be fully leveraged by neuromorphic hardware which adopts distributing memory and self-contained computational units in decentralized many-

core architecture, our model provides a templet promoting the graphic application for neuromorphic computing.

4 Conclusion

In this work, we report a general SNN framework for graph-structured data with an iterative spiking message passing method. By proposing the STFN method, we incorporate the graph propagation and spiking dynamics into one unified paradigm and reconcile them in a collaborative mode. Our framework is flexible and transferable for various propagation operations and scenarios, and we instantiate it into two specific models for demonstrations: a GC-SNN for graph convolution and a GA-SNN for graph attention. The experimental results on three benchmark datasets demonstrate the model effectiveness and powerful representation capability. More importantly, Graph SNNs possess high-efficiency advantages conducive to the implementation on neuromorphic hardware and applications for graphic scenarios. Overall, this work sheds new light on spiking dynamics research interacted with graphic topology, which may facilitate the understanding for advanced cognitive intelligence.

Acknowledgments

References

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

[Cancan, 2019] Murat Cancan. On ev-degree and ve-degree topological properties of tickysim spiking neural network. *Computational Intelligence and Neuroscience*, 2019, 2019.

[Cao *et al.*, 2015] Zhiqiang Cao, Long Cheng, Chao Zhou, Nong Gu, Xu Wang, and Min Tan. Spiking neural network-based target tracking control for autonomous mobile robots. *Neural Computing and Applications*, 26(8):1839–1847, 2015.

[Defferrard *et al.*, 2016] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *arXiv preprint arXiv:1606.09375*, 2016.

[Getoor, 2005] Lise Getoor. Link-based classification. In *Advanced methods for knowledge discovery from complex data*, pages 189–207. Springer, 2005.

[Gilmer *et al.*, 2017] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1263–1272, 2017.

[Gu *et al.*, 2020] Fuqiang Gu, Weicong Sng, Tasbolat Taunyazov, and Harold Soh. Tactilesgnet: A spiking graph neural network for event-based tactile object recognition. *arXiv preprint arXiv:2008.08046*, 2020.

[Haessig *et al.*, 2019] Germain Haessig, Xavier Berthelon, Sio-Hoi Ieng, and Ryad Benosman. A spiking neural network model of depth from defocus for event-based neuromorphic vision. *Scientific reports*, 9(1):1–11, 2019.

[Hamilton *et al.*, 2017] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034, 2017.

[Hamilton *et al.*, 2020] Kathleen Hamilton, Tiffany Mintz, Prasanna Date, and Catherine D Schuman. Spike-based graph centrality measures. In *International Conference on Neuromorphic Systems 2020*, pages 1–8, 2020.

[Ioffe and Szegedy, 2015] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[Jovanović and Rotter, 2016] Stojan Jovanović and Stefan Rotter. Interplay between graph topology and correlations of third order in spiking neuronal networks. *PLoS computational biology*, 12(6):e1004963, 2016.

[Kipf and Welling, 2019] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. 2019.

[Perozzi *et al.*, 2014] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[Piekniewski, 2007] Filip Piekniewski. Emergence of scale-free graphs in dynamical spiking neural networks. In *2007 International Joint Conference on Neural Networks*, pages 755–759. IEEE, 2007.

[Sala and Cios, 1999] Dorel M Sala and Krzysztof J Cios. Solving graph algorithms with networks of spiking neurons. *IEEE transactions on neural networks*, 10(4):953–957, 1999.

[Silva *et al.*, 2017] Marco Silva, Marley MBR Vellasco, and Edson Cataldo. Evolving spiking neural networks for recognition of aged voices. *Journal of voice*, 31(1):24–33, 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.

[Whittington *et al.*, 2020] James CR Whittington, Timothy H Muller, Shirley Mark, Guifen Chen, Caswell Barry, Neil Burgess, and Timothy EJ Behrens. The tolman-eichenbaum machine: Unifying space and relational memory through generalization in the hippocampal formation. *Cell*, 183(5):1249–1263, 2020.

[Wu *et al.*, 2018] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 12:331, 2018.

[Wu *et al.*, 2019] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, Yuan Xie, and Luping Shi. Direct training for spiking neural networks: Faster, larger, better. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1311–1318, 2019.

[Yang *et al.*, 2016] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR, 2016.

[Zheng *et al.*, 2020] Hanle Zheng, Yujie Wu, Lei Deng, Yifan Hu, and Guoqi Li. Going deeper with directly-trained larger spiking neural networks. *arXiv preprint arXiv:2011.05280*, 2020.

[Zhou *et al.*, 2018] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.

[Zhu *et al.*, 2020] Lin Zhu, Siwei Dong, Jianing Li, Tiejun Huang, and Yonghong Tian. Retina-like visual image reconstruction via spiking neural model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1438–1446, 2020.