

Spiking Heterogeneous Graph Attention Networks

Buqing Cao¹, Qian Peng^{2,*}, Xiang Xie³, Liang Chen⁴, Min Shi⁵, Jianxun Liu³

¹School of Computer Science and Artificial Intelligence, Hunan University of Technology, Zhuzhou, China

²School of Computer Science and Engineering, Central South University, Changsha, China

³School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan, China

⁴School of Data and Computer, Sun Yat-Sen University, Guangzhou, China

⁵School of Computing & Informatics, University of Louisiana at Lafayette, LA, USA

{buqingcao, pengqian369, xiangxie001, ljsx529}@gmail.com, chenliang6@mail.sysu.edu.cn, min.shi@louisiana.edu

Abstract

Real-world graphs or networks are usually heterogeneous, involving multiple types of nodes and relationships. Heterogeneous graph neural networks (HGNNs) can effectively handle these diverse nodes and edges, capturing heterogeneous information within the graph, thus exhibiting outstanding performance. However, most methods of HGNNs usually involve complex structural designs, leading to problems such as high memory usage, long inference time, and extensive consumption of computing resources. These limitations pose certain challenges for the practical application of HGNNs, especially for resource-constrained devices. To mitigate this issue, we propose the Spiking Heterogeneous Graph Attention Networks (SpikingHAN), which incorporates the brain-inspired and energy-saving properties of Spiking Neural Networks (SNNs) into heterogeneous graph learning to reduce the computing cost without compromising the performance. Specifically, SpikingHAN aggregates metapath-based neighbor information using a single-layer graph convolution with shared parameters. It then employs a semantic-level attention mechanism to capture the importance of different metapaths and performs semantic aggregation. Finally, it encodes the heterogeneous information into a spike sequence through SNNs, simulating bioinformatic processing to derive a binarized 1-bit representation of the heterogeneous graph. Comprehensive experimental results from three real-world heterogeneous graph datasets show that SpikingHAN delivers competitive node classification performance. It achieves this with fewer parameters, quicker inference, reduced memory usage, and lower energy consumption.

Introduction

Graph Neural Networks (GNNs) are excellent in combining graph data structures and node features, and have been widely used in various domains (Liu et al. 2024; Zhang, Li, and Nejdli 2024; Wu et al. 2024; Peng et al. 2025). Most GNNs are designed to learn node embedding vectors in homogeneous graphs, which consist of a single type of nodes and edges. However, real-world entity nodes and interaction edges often consist of multiple types, thereby forming heterogeneous graphs with rich structural and semantic information. Heterogeneous Graph Neural Networks

(HGNNs) are specifically designed to handle heterogeneous graphs (Shi et al. 2016), leveraging meta-paths to capture the complex relationships among various types of nodes and edges. This not only improves the expression capability of the model but also enhances its adaptability to complex networks, showing great potential in fields such as social network analysis, recommendation systems, and bioinformatics (Ma et al. 2024; Cao et al. 2024; Peng et al. 2023). For instance, in the ACM paper dataset illustrated in Fig. 1(a), the heterogeneous graph consists of three types of nodes and two types of edges. Using two predefined meta-paths, PAP and PSP, relationships between papers can be uncovered, such as papers sharing the same authors or papers belonging to the same topic.

Despite the excellent performance of HGNNs, many existing methods rely on complex structural designs, leading to problems challenges such as high memory usage, significant computational resource demands, lengthy inference time, and excessive power consumption during training. These limitations pose challenges for the deployment and expansion of HGNNs in practical applications, especially for resource-constrained devices (Zhou et al. 2023). For example, Fig. 1(b) illustrates the hierarchical aggregation paradigm employed by the widely-used HGNNs called HAN (Wang et al. 2019). HAN assigns a distinct node attention module for each meta-path, resulting in a significant increase in the model’s parameters, memory usage, and computational resource requirements as the number of meta-paths grows. Although this multi-level attention mechanism can effectively capture complex heterogeneous relationships, it intensifies the burden of computation and storage. Inspired by the brain’s information processing methods, spiking neural networks (SNNs) use event-triggered and time-driven signals to update the parameters of neuron nodes, and can be categorized as brain-inspired networks characterized by discretization and sparsity (Baronig et al. 2025; Feng et al. 2022). Different from traditional neural networks that pass messages through the neurons with floating-point value, the neurons of SNNs communicate in a sparse and binarized manner. These characteristics make SNNs highly suitable for application in low-energy consumption scenarios, mobile devices, and other resource-constrained environments.

Inspired by the successful application of SNNs in com-

*Corresponding authors.

Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

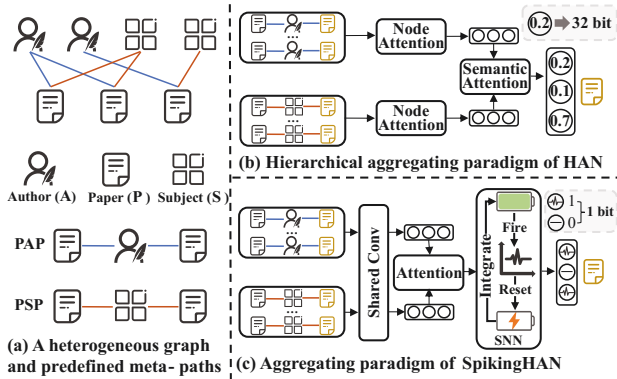


Figure 1: Diagram of a heterogeneous graph and comparison between our model and HAN

puter vision (Kim, Chough, and Panda 2022), researchers began to extend SNNs to graph data. Recent studies (Li et al. 2023a; Zhu et al. 2022; Li et al. 2023b; Yin et al. 2024) have integrated SNNs with graph convolutional networks, graph contrastive learning, and dynamic graphs, achieving competitive performance with reduced computational costs. These works also provide theoretical guarantees, demonstrating that binary spike-based graph networks possess expressive capabilities comparable to floating-point GNNs. This integration not only reduces energy consumption and computing resource usage, but also brings new possibilities for applying GNNs in resource-constrained environments. Although the application of SNNs to GNNs has gained increasing attention, their potential in heterogeneous graphs—a more prevalent graph data scenario in real-world applications—has yet to be fully explored and studied. Therefore, this paper fills this gap by designing effective SNNs for heterogeneous graphs.

This paper proposes a novel spiking heterogeneous graph attention network, called SpikingHAN, with its simplified aggregation paradigm illustrated in Fig. 1(c). SpikingHAN introduces SNNs into the learning process of heterogeneous graphs and constructs binary heterogeneous graph representation in a compact and efficient manner. Unlike traditional HGNNs that require distinct aggregation mechanisms for each meta-path, SpikingHAN employs a single-layer graph convolution with shared parameters to perform aggregation for each meta-path, thereby significantly reducing the complexity of the model while maintaining the necessary expressive capability. SpikingHAN further uses the semantic-level attention mechanism to learn the importance weights of different meta-paths and performs targeted semantic aggregation based on these weights. Then, the Integrate, Fire, and Reset events of the SNNs are used to aggregate and update the spike signals at each time step. This process learns a sparse and effective binary representation, fully leveraging the advantages of SNNs for heterogeneous graph data and enabling fast inference. To conclude, the major contributions of this paper are summarized as follows:

- A novel spiking heterogeneous graph attention network SpikingHAN is proposed, which innovatively combines

SNNs and HGNNs. By simulating the spiking mechanism of biological neurons, it achieves low-energy and efficient computation, providing a new solution for the efficient processing of heterogeneous graphs.

- To our knowledge, SpikingHAN is the first attempt to integrate SNNs into heterogeneous graph data. Our work enables SNNs to be directly applied to heterogeneous graphs, especially in low-energy and resource-constrained environments, further promoting the development of heterogeneous graph-based applications.
- Comprehensive experiments on three real-world datasets indicate that, compared with similar models that output floating-point values, SpikingHAN achieves competitive node classification performance with fewer parameters, faster inference, smaller memory usage, and lower energy consumption.

Related Work

Heterogeneous Graph Neural Network

Existing HGNNs are generally categorized into metapath-based and relation-based models. Metapath-based models maintain the heterogeneity of heterogeneous graphs by constructing neighbors using predefined meta-paths. MAGNN (Fu et al. 2020) transforms node content features, incorporates intermediate nodes, and integrates multi-metapath semantics. THGNN (Xu et al. 2021) extracts fine-grained topic-aware semantics by decomposing topics and leveraging global textual knowledge, improving link prediction and interpretability. HOAE (Li et al. 2024) applies a transformer-based mechanism to enhance attribute learning from heterogeneous neighbors and perform high-order attribute extraction via meta-paths. PHGT (Lu et al. 2024) captures high-order semantics and long-range dependencies using node, semantic, and global tokens. In contrast, relation-based models avoid manual meta-paths by directly aggregating neighbor relation features. GTN (Yun et al. 2019) adaptively learns edge types and composite relations without requiring domain knowledge. HetSANN (Hong et al. 2020) processes multi-relation information via projection and attention, without predefined meta-paths. ie-HGCN (Yang et al. 2021) automatically selects useful meta-paths through hierarchical aggregation, reducing preprocessing and computation. DAHGN (Zhao and Jia 2024) addresses degree bias via dual-view contrastive learning between heterogeneous and homogeneous graph views. Although HGNNs perform well, most methods involve complex structural design, leading to high memory usage, long inference times, and large computing resource consumption. These limitations pose challenges for practical applications, especially on resource-constrained devices.

Spiking Neural Networks

Spiking Neural Networks (SNNs) bridge neuroscience and machine learning by simulating biological neurons with sparse, event-driven spikes (Tavanaei et al. 2019). Due to their biologically plausible mechanisms and low energy consumption, SNNs are promising for energy-efficient applications. In computer vision, SNNs have shown strong

potential. Spiking Transformer (Zhou et al. 2022) combines SNNs with self-attention, achieving state-of-the-art performance on ImageNet with improved energy efficiency. Spiking-YOLO (Kim et al. 2020) introduces channel normalization and symbol neurons for accurate object detection in deep SNNs. (Kim, Chough, and Panda 2022) re-designed semantic segmentation architectures like FCN and DeepLab using SNNs with surrogate gradient training, enhancing robustness and energy efficiency. Inspired by these successes, researchers began exploring SNNs on graph data. On homogeneous graphs, SpikingGCN (Zhu et al. 2022) integrates GCNs and SNNs, encoding graph structures into spike sequences and achieving excellent performance with energy efficiency on neuromorphic chips. GSAT (Wang and Jiang 2022) generates sparse attention coefficients to enhance noise robustness in graph edge structures. SpikeGCL (Li et al. 2023b) introduces SNNs into graph contrastive learning, producing efficient binarized 1-bit representations with low storage cost. On dynamic graphs, SpikeNet (Li et al. 2023a) replaces traditional RNNs with spiking neurons to capture structural evolution efficiently. (Yin et al. 2024) further improve dynamic graph representation by propagating early-layer information directly to the final layer and applying implicit differentiation to reduce memory usage. Although the application of SNNs to graph data has gradually attracted attention, SNNs have not been fully valued and studied in heterogeneous graphs, which are common in real-world scenarios. To address this, our proposed SpikingHAN applies SNNs to heterogeneous graphs and achieves quite competitive performance in node classification tasks with fewer parameters, faster inference, smaller memory usage, and lower energy consumption.

Preliminary

Definition 1: Heterogeneous Graph. A heterogeneous graph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where \mathcal{V} and \mathcal{E} represent the sets of nodes and edges, respectively. The heterogeneous graph \mathcal{G} is also associated with a node type mapping function $\phi : \mathcal{V} \rightarrow \mathcal{A}$ and an edge type mapping function $\psi : \mathcal{E} \rightarrow \mathcal{R}$. \mathcal{A} and \mathcal{R} represent predefined sets of node types and edge types, respectively, where $|\mathcal{A}| + |\mathcal{R}| > 2$.

Definition 2: Metapath-based Neighbor. A meta-path Φ is a path of the form $A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \dots \xrightarrow{R_l} A_{l+1}$, which describes the composite relationship $R = R_1 \circ R_2 \circ \dots \circ R_l$ between the node types A_1 and A_{l+1} , where \circ denotes the composite operator on this relationship. Given a meta-path Φ in a heterogeneous graph, the metapath-based neighbors N_i^Φ of node i are the set of nodes connected to node i through the meta-path Φ . If the meta-path Φ is symmetric, then N_i^Φ includes node i itself.

Definition 3: Spiking Neural Network. SNNs usually have three core characteristics: (1) Integrate. Spiking neurons accumulate current through capacitors, gradually increasing the charge; (2) Fire. When the membrane potential reaches or exceeds the threshold V_{th} , the neuron generates a spike signal; and (3) Reset. After the spike signal is generated, the membrane potential is reset. There are generally

two reset methods (Rueckauer et al. 2016), one is to reset the membrane potential to a constant V_{reset} (usually 0, and $V_{reset} < V_{th}$), and the other is to reset by subtracting the threshold V_{th} . The formal descriptions of these three core characteristics are respectively shown in equations (1), (2), and (3),

$$\text{Integrate} : V^t = \Psi(V^{t-1}, I^t) \quad (1)$$

$$\text{Fire} : S^t = \Theta(V^t - V_{th}) \quad (2)$$

$$\text{Reset} : V^t = \begin{cases} S^t \cdot V_{reset} + (1 - S^t) \cdot V^t \\ S^t \cdot (V^t - V_{th}) + (1 - S^t) \cdot V^t \end{cases} \quad (3)$$

where I^t and V^t represent the input current and membrane potential at time step t , respectively. Whether a spiking neuron generates a spike signal is determined by the Heaviside function $\Theta(\cdot)$, when $x \geq 0$, $\Theta(x) = 1$, otherwise $\Theta(x) = 0$. The function $\Psi(\cdot)$ is used to describe how spiking neurons receive input current and accumulate membrane potential. We can represent $\Psi(\cdot)$ using the Integrate-and-Fire (IF) model (Salinas and Sejnowski 2002) and its variant the Leaky Integrate-and-Fire (LIF) model (Gerstner et al. 2014), as shown in equations (4) and (5),

$$\text{IF} : V^t = V^{t-1} + I^t \quad (4)$$

$$\text{LIF} : V^t = V^{t-1} + \frac{1}{\tau_m} (I^t - (V^{t-1} - V_{th})) \quad (5)$$

where τ_m is the membrane time constant used to control the rate of decay of the membrane potential, it usually needs to be adjusted manually. In the Parametric LIF (PLIF) model (Fang et al. 2021), τ_m can be automatically optimized during the training process to capture and learn different neuronal dynamics. This paper uses the surrogate gradient method to define $\Theta'(x) \triangleq \sigma'(\alpha x)$ during the loss backpropagation process (Li et al. 2023a), where $\sigma(\cdot)$ is the activation function, and α is the smoothing factor.

Methodology

This section introduces a novel heterogeneous graph neural network termed SpikingHAN. SpikingHAN tackles the challenge of semi-supervised node classification in a brain-inspired and energy-efficient manner. Its overall framework is shown in Fig. 2, consisting of three main components, including the Shared Convolution and Attention, Fully Connected Layer and SNN, and Spikes Statistic and Prediction.

Shared Convolution and Attention

As one of the most representative models in GNNs, GCN (Kipf 2016) is used in SpikingHAN to construct the shared graph convolution module that helps to aggregate each node’s metapath-based neighbor information. Previous work usually assigns independent aggregation modules to each meta-path. Although this approach is effective, it may result in a significant increase in model complexity and is prone to overfitting problems when the number of convolution layers increases. In contrast, SpikingHAN uses shared parameters to perform metapath-based neighbors aggregation operations and executes only one layer of convolution oper-

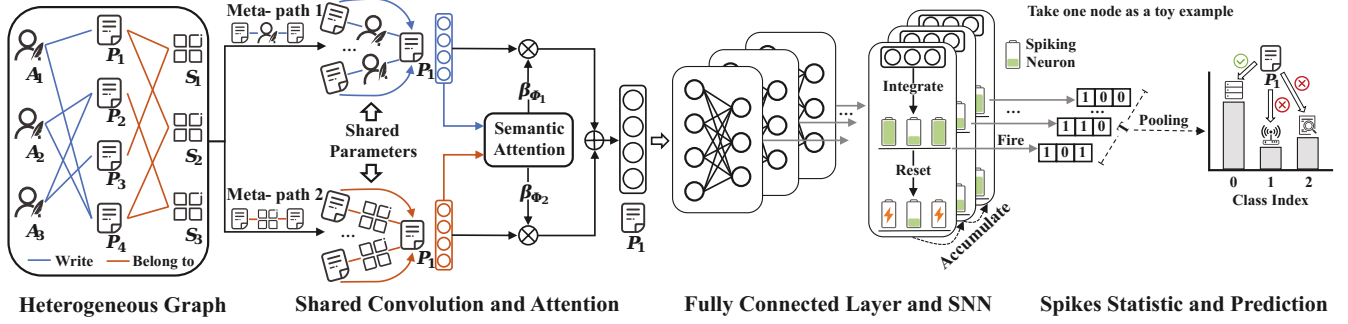


Figure 2: The overall framework of SpikingHAN

ations, thus minimizing the model complexity while maintaining its expressive power. The single-layer graph convolution with shared parameters of SpikingHAN is formalized as shown in equation (6),

$$h_i^{\Phi_p} = \sigma \left(\sum_{j \in N_i^{\Phi_p}} \frac{h_j^0 \cdot W_1}{\sqrt{\tilde{D}_i \cdot \tilde{D}_j}} \right) \quad (6)$$

where $N_i^{\Phi_p}$ is the set of neighbor nodes of node i based on the meta-path $\Phi_p \in \{\Phi_1, \Phi_2, \dots, \Phi_P\}$. $\sigma(\cdot)$ is the activation function. h_j^0 is the initial feature representation of node j . $W_1 \in \mathbb{R}^{d_{in} \times d_{hd}}$ is the learnable weight matrix, d_{in} and d_{hd} are the dimension of node initial feature and the hidden layer dimension respectively. $h_i^{\Phi_p}$ is the new feature representation of node i after the graph convolution operation with shared parameters. \tilde{D}_i and \tilde{D}_j are the degrees of node i and node j respectively. Given a set of meta-paths $\{\Phi_1, \Phi_2, \dots, \Phi_P\}$, the initial node features will generate P sets of node embedding $\{h^{\Phi_1}, h^{\Phi_2}, \dots, h^{\Phi_P}\}$ with specific meta-path semantics after passing through the graph convolution with shared parameters.

After aggregating metapath-based neighbor information for each node, selectively integrating the semantics captured by different meta-paths is essential for learning more comprehensive node embeddings. Inspired by the semantic-level attention mechanism in (Wang et al. 2019), we adopt this attention mechanism to automatically determine the importance of various meta-paths and integrate them into their corresponding semantics. Specifically, the node embeddings of specific meta-path semantics are transformed nonlinearly, and their importance is measured by calculating the similarity between the transformed embeddings and the semantic-level attention vector q , formalized as shown in equation (7),

$$I_{\Phi_p} = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} q^T \cdot \tanh(h_i^{\Phi_p} \cdot W_2 + b) \quad (7)$$

where $W_2 \in \mathbb{R}^{d_{hd} \times d_{hd}}$ is the learnable weight matrix, b is the bias vector. The importance weight β_{Φ_p} of the meta-path Φ_p can be obtained by applying Softmax normalization to the importance of all meta-paths, as shown in equation (8).

$$\beta_{\Phi_p} = \frac{\exp(I_{\Phi_p})}{\sum_{p=1}^P \exp(I_{\Phi_p})} \quad (8)$$

The learned importance weight β_{Φ_p} can also serve as the attention coefficient for the meta-path to selectively integrate the various semantics revealed by the meta-path, as shown in equation (9),

$$H = \sum_{p=1}^P \beta_{\Phi_p} \cdot h^{\Phi_p} \quad (9)$$

where H represents the node embedding that integrates various meta-path semantics, and P is the number of meta-paths.

Fully Connected Layer and SNN

The fully connected layer processes node embeddings H , which incorporate various meta-path semantics, and uses a trainable weight matrix to transform the high-dimensional feature space into a low-dimensional feature space. These low-dimensional features are subsequently used as input currents to the SNN. In the SNN, the input currents charge the spiking neurons, which then trigger a series of operations such as charge accumulation, spike fire and membrane potential reset.

Most deep SNN models use a multi-layer network structure that combines linear and nonlinear functions to process the input. However, based on the assumption of LightGCN (He et al. 2020), the depth of SNN is not a critical factor for predicting unknown labels on graphs. For this reason, we retain only the linear transformations in the fully connected layer and remove redundant modules such as nonlinear activation functions and biases to simplify the model and improve inference speed. The result $H \cdot W_3$ after linear transformation will be used as the input of SNN. First, the Integrate operation is performed, that is, charging the spiking neurons so that the charge gradually accumulates, as shown in equation (10),

$$V^t = V^{t-1} + \frac{1}{\tau_m} (\text{dropout}(H \cdot W_3) - (V^{t-1} - V_{th})) \quad (10)$$

where $t = 1, 2, \dots, T$, V^t is the membrane potential at time step t , $V^0 = 0$. τ_m is a learnable parameter used to control the decay rate of membrane potential. $W_3 \in \mathbb{R}^{d_{hd} \times d_{out}}$ is the learnable weight matrix, d_{out} is set to the number of classes of the target node.

When the accumulated membrane potential V^t of the spiking neuron reaches or exceeds the given threshold V_{th} ,

the spiking neuron generates a spike through the Heaviside step function $\Theta(\cdot)$, i.e., the Fire operation, as shown in equation (11).

$$\Theta(V^t) = \begin{cases} 1, & V^t \geq V_{th} \\ 0, & V^t < V_{th} \end{cases} \quad (11)$$

After a spiking neuron generates a spike, its membrane potential will release its potential like a biological neuron and begin accumulating voltage again, i.e., the Reset operation. In SpikingHAN, the membrane potential is reset by subtracting the threshold, as shown in equation (12).

$$V^t = \Theta(V^t) \cdot (V^t - V_{th}) + (1 - \Theta(V^t)) \cdot V^t \quad (12)$$

Spikes Statistic and Prediction

The spike sequence of each node at time step t is generated by the spiking neurons. By applying average pooling on the outputs of the spiking neurons over multiple time steps, the class firing rate for each node can be obtained. This class firing rate will be used as the probability for the final node classification prediction, as shown in equation (13),

$$\hat{y} = \frac{1}{|T|} \sum_{t=1}^T \Theta(V^t) \quad (13)$$

where \hat{y} is the probability of the model classification prediction.

For the semi-supervised node classification task, this paper minimizes the cross-entropy loss through back propagation and gradient descent to optimize the weight parameters of the model. This cross-entropy loss of semi-supervised learning is shown in equation (14),

$$\mathcal{L} = - \sum_{i \in \mathcal{V}_L} y_i \cdot \ln(\hat{y}_i) \quad (14)$$

where \mathcal{V}_L is the set of labeled node indices. y_i is the one-hot encoding of the true label for node i . \hat{y}_i is the probability of node i classification prediction.

Experiment

To validate the effectiveness of SpikingHAN, we perform comprehensive experiments on three public real-world datasets, aiming to answer the following research questions:

- RQ1: How does SpikingHAN perform in node classification?
- RQ2: Does SpikingHAN have an advantage in terms of computational cost?
- RQ3: How do different configurations of spiking neurons and time steps affect the performance of SpikingHAN?

Experimental Settings

Dataset description. The experiments employ three commonly used heterogeneous graph datasets to evaluate the performance of SpikingHAN. The detailed characteristics of the datasets are summarized in Table 1.

Datasets	Nodes	Edges	Meta-paths
DBLP	Paper (P): 14,328	A-P: 19,645	APA
	Author (A): 4,057	P-V: 14,328	APVPA
	Venue (V): 20	P-T: 85,810	APTPA
	Term (T): 7,723		
ACM	Paper (P): 3,025	P-A: 9,744	PAP
	Author (A): 5,825	P-S: 3,025	PSP
	Subject (S): 26		
IMDB	Movie (M): 4,278	M-D: 4,278	MDM
	Director (D): 2,081	M-A: 12,828	MAM
	Actor (A): 5,257		

Table 1: The description of datasets

Baselines. SpikingHAN will be compared with three types of GNN models: (1) homogeneous GNNs, such as GAT (Veličković et al. 2017) and DAGNN (Liu, Gao, and Ji 2020). (2) homogeneous graph SNNs, such as SpikingGCN (Zhu et al. 2022) and SpikeGCL (Li et al. 2023b). and (3) heterogeneous GNNs, such as HAN (Wang et al. 2019), HINormer (Mao et al. 2023), and PHGT (Lu et al. 2024).

For more details on the datasets, model implementation, and baselines, please refer to the code link <https://github.com/QianPeng369/SpikingHAN>.

Performance Comparison (RQ1)

Table 2 summarizes the classification results across different datasets and training ratios. By analyzing the experimental results, we arrive at the following conclusions:

- Heterogeneous GNNs typically outperform homogeneous GNNs and spiking GNNs on tasks involving complex structures and diverse node types, as they can model multi-type relationships and capture richer semantic information. In contrast, homogeneous models treat all nodes and edges uniformly, making it difficult to distinguish heterogeneous semantics.
- The homogeneous graph SNNs (e.g., SpikingGCN and SpikeGCL) are much smaller than the traditional homogeneous GNNs (e.g., GAT and DAGNN) in terms of the number of model parameters, but there is no significant gap in performance. This indicates that introducing SNNs into GNNs can effectively reduce model complexity while maintaining competitive performance, further demonstrating the application value of spiking neural networks in graph neural networks.
- Combined with Table 3, Fig. 3, and Fig. 4, it can be observed that traditional heterogeneous GNNs (e.g., HINormer and PHGT) achieve strong performance but incur high computational costs. In contrast, SpikingHAN significantly decreases the computing cost while still achieving or even partially exceeding the performance of these traditional methods. This advantage is due to the fact that SpikingHAN simulates the biological neuron spike firing mechanism to achieve low-energy and efficient computation.

Dataset	Metric	Tr. ratio	GAT	DAGNN	SpikingGCN	SpikeGCL	HAN	HINormer	PHGT	SpikingHAN
DBLP	Mi-F1	20%	91.4±0.4	91.9±0.7	88.7±0.6	90.6±0.2	93.1±0.1	93.6±0.1	93.7±0.2	93.7±0.1
		40%	91.9±0.3	91.6±0.9	90.4±0.3	91.3±0.3	93.6±0.1	94.2±0.2	94.6±0.2	93.8±0.2
		60%	92.9±0.4	91.8±0.9	90.3±0.3	91.5±0.3	93.7±0.2	93.7±0.3	94.3±0.2	93.7±0.1
	Ma-F1	20%	90.8±0.4	91.2±0.8	88.1±0.6	89.9±0.2	92.5±0.1	92.2±0.1	93.3±0.2	93.2±0.1
		40%	91.4±0.3	91.1±1.0	89.5±0.3	90.7±0.4	93.1±0.1	93.8±0.2	94.2±0.2	93.3±0.1
		60%	91.6±0.4	91.1±1.0	89.3±0.4	90.9±0.4	93.2±0.2	93.2±0.4	93.6±0.1	93.2±0.2
ACM	Mi-F1	20%	91.1±0.4	91.5±0.3	91.2±0.4	89.4±0.2	92.8±0.2	93.2±0.2	93.4±0.1	93.3±0.1
		40%	92.1±0.2	92.2±0.4	91.5±0.2	90.2±0.1	93.3±0.2	93.5±0.2	93.5±0.1	93.0±0.3
		60%	92.1±0.2	92.3±0.3	91.9±0.2	90.6±0.2	93.1±0.3	93.2±0.1	93.3±0.2	92.9±0.2
	Ma-F1	20%	91.2±0.4	91.2±0.4	91.2±0.5	89.4±0.2	92.9±0.2	93.1±0.2	93.1±0.1	93.4±0.1
		40%	92.1±0.2	92.0±0.4	91.4±0.1	90.2±0.1	93.3±0.2	93.4±0.2	93.4±0.1	93.1±0.3
		60%	92.1±0.2	92.2±0.3	91.8±0.2	90.5±0.2	93.1±0.3	93.2±0.2	93.1±0.2	92.9±0.2
IMDB	Mi-F1	20%	58.5±0.3	59.4±0.2	51.3±0.6	55.2±0.4	61.3±0.4	63.3±0.2	63.2±0.2	62.9±0.2
		40%	61.9±0.3	61.7±0.5	54.4±0.8	58.6±0.3	63.1±0.4	64.3±0.3	64.5±0.1	64.2±0.3
		60%	62.5±0.3	61.9±0.6	55.7±0.9	60.5±0.4	64.6±0.5	64.8±0.2	65.0±0.2	65.2±0.5
	Ma-F1	20%	58.3±0.4	59.1±0.2	50.4±0.7	54.7±0.4	61.1±0.4	62.9±0.2	63.0±0.2	62.7±0.1
		40%	61.7±0.4	61.3±0.6	54.1±0.9	58.2±0.3	62.8±0.3	63.8±0.2	64.1±0.1	63.7±0.4
		60%	62.1±0.4	61.2±0.6	55.3±0.9	60.2±0.4	64.3±0.5	64.4±0.2	64.6±0.2	65.1±0.4

Table 2: Experimental results for node classification (%). Results are the mean and standard deviation obtained by running 10 random seeds. The best result in each row is highlighted in bold. (Tr. Ratio: Training ratio, Mi-F1: Micro-F1, Ma-F1: Macro-F1)

Method	DBLP		ACM		IMDB	
	Param	Memory	Param	Memory	Param	Memory
GAT	349,196	963.37	960,521	1994.92	1,572,873	1174.64
DAGNN	43,400	917.80	239,878	1153.13	785,926	627.71
HAN	292,868	1,676.8	1,985,283	542.39	6,419,715	450.17
HINormer	7,348,964	6361.18	3,633,287	1093.61	8,577,159	2108.72
PHGT	8,791,360	7156.53	5,579,065	2414.34	9,311,232	4459.98
SpikingHAN	15,201	45.2	128,385	137.32	102,593	220.19

Table 3: Number of model parameters and maximum GPU memory allocation (MB) during training

Runtime Complexity (RQ2)

We further recorded the runtime complexity of SpikingHAN and partial baseline methods under different datasets, including the number of model parameters, the maximum GPU memory allocation during training, training time, and GPU energy consumption per epoch. These data are obtained through GPU monitoring and management library pynvml provided by NVIDIA, and are averaged based on the results of 10 different random seeds, as shown in Table 3, Fig. 3, and Fig. 4. It can be seen from the data results that the number of parameters for SpikingHAN is significantly smaller than the other methods, while the number of parameters of heterogeneous graph neural network methods with excellent performance (e.g., HINormer and PHGT) is as high as mil-

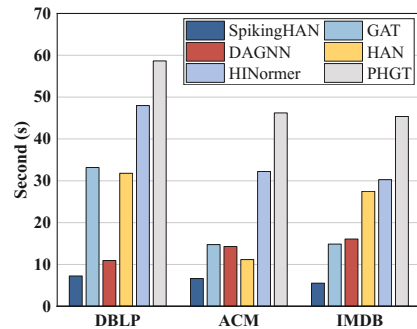


Figure 3: Model training time on different datasets

lions. SpikingHAN achieves quite competitive performance with fewer parameters, which demonstrates that SpikingHAN is more efficient in terms of model complexity, leading to faster training speed and lower overfitting risk. At the same time, SpikingHAN has the lowest maximum GPU memory allocation on all datasets, and this low memory requirement makes SpikingHAN more advantageous when memory resources are limited. In addition, Fig. 4 shows the GPU energy consumption per epoch during training on the three datasets. It can be observed that the energy consumption for the first epoch is higher compared to subsequent epochs. This is mainly because the first epoch usually involves model initialization and weight setting, while the sub-

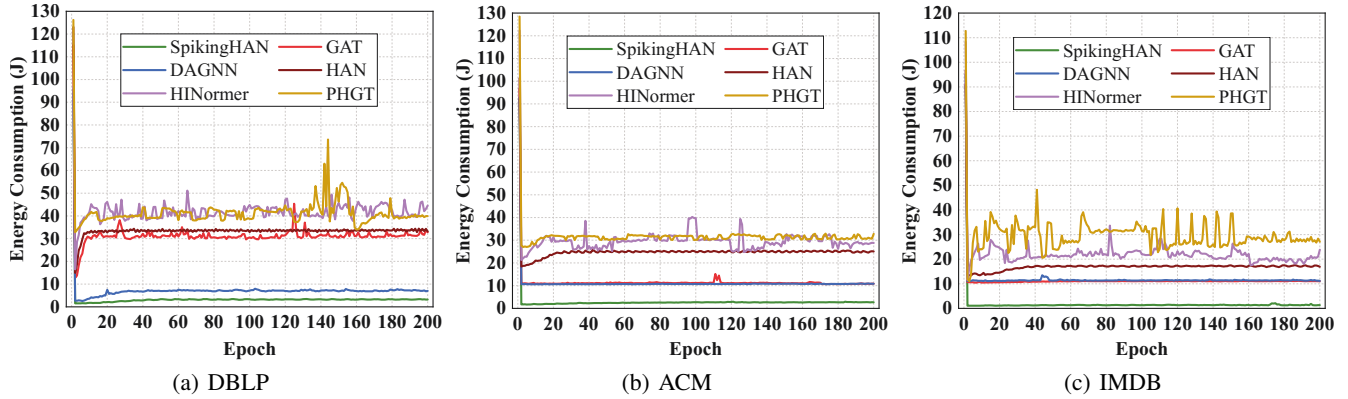


Figure 4: GPU energy consumption per epoch during model training

sequent epochs are mainly fine-tuning. Although PHGT performs best in classification effect, its GPU energy consumption is approximately 13 times that of SpikingHAN, and the performance improvement is not significant. The overall energy consumption of SpikingHAN on the three datasets is the lowest and relatively stable among all the compared methods. This demonstrates that SpikingHAN not only has excellent performance in dealing with heterogeneous graph learning tasks, but also can effectively reduce energy consumption, which has important economic and environmental value for large-scale applications and practical deployment.

Sensitivity Analysis (RQ3)

This section further studies how different spike neurons and time steps T affect the performance of SpikingHAN. From the experimental results, we conclude the following:

- **The impact of spiking neurons.** We use IF (Salinas and Sejnowski 2002), LIF (Gerstner et al. 2014), and PLIF (Fang et al. 2021) to construct spiking neurons for SNNs, and conduct experiments with a training rate of 20% on different datasets. Experimental results show that the simple IF neuron is already capable of enabling SpikingHAN to achieve good performance. The LIF neuron improves the performance by adding biologically reasonable leakage terms to IF neuron. The PLIF neuron sets the leakage term in the LIF neuron as a learnable parameter, which confers better flexibility and biological plausibility to the SNNs. Therefore, PLIF performs slightly better than LIF in most cases.
- **The impact of time steps T .** We investigate the effect of time steps T on the performance of SpikingHAN on different datasets. In fact, the best T value for each dataset is not the global optimum, but a local optimum that balances computing cost and performance. This is because as the time steps T increases, the frequency of spiking neurons performing Integrate, Fire, and Reset operations will also increase, and the performance of SpikingHAN may improve, but the running time and memory consumption will also increase, resulting in higher computational costs. Therefore, selecting time steps T should bal-

ance computing cost and model performance, and should be considered in combination with the specific requirements of downstream tasks.

Conclusion

In this paper, we propose a novel Spiking Heterogeneous Graph Attention Network model termed SpikingHAN, which incorporates spiking neural networks—known for their brain-inspired and energy-efficient properties—into heterogeneous graph learning, aiming to reduce computational costs while preserving performance. SpikingHAN aggregates metapath-based neighbor nodes through a single-layer graph convolution with shared parameters, and utilizes the semantic-level attention mechanism to aggregate different meta-path semantics. Finally, SNN is used to simulate the spike firing mechanism of biological neurons, encoding heterogeneous information into spike sequences. The pooled spike sequences are then used for prediction, enabling efficient computing with low energy consumption. Experimental results on three real-world datasets indicate that SpikingHAN achieves performance competitive with the best baseline methods using a binary 1-bit representation. At the same time, compared with other heterogeneous graph neural network methods, SpikingHAN shows efficiency advantages in training speed, model parameters, memory usage, and energy consumption. From the perspective of building environment-friendly machine learning models, our work highlights the potential of spiking neural networks as a promising direction for sustainable heterogeneous graph learning. The energy-efficient, event-driven nature of spiking computation enables effective graph representation learning with significantly reduced resource requirements, offering new insights into the design of green and scalable graph learning architectures.

Acknowledgments

The work is supported by the National Natural Science Foundation of China (No. 62572186) and the Open Project Funding of the Key Laboratory of Intelligent Sensing System and Security (Hubei University), Ministry of Education.

References

- Baronig, M.; Ferrand, R.; Sabathiel, S.; and Legenstein, R. 2025. Advancing spatio-temporal processing through adaptation in spiking neural networks. *Nature Communications*, 16(1): 5776.
- Cao, B.; Peng, Q.; Xie, X.; Peng, Z.; Liu, J.; and Zheng, Z. 2024. Web service recommendation via combining topic-aware heterogeneous graph representation and interactive semantic enhancement. *IEEE Transactions on Services Computing*, 17(6): 4451–4466.
- 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 *Proceedings of the IEEE/CVF international conference on computer vision*, 2661–2671.
- Feng, L.; Liu, Q.; Tang, H.; Ma, D.; and Pan, G. 2022. Multi-level firing with spiking ds-resnet: Enabling better and deeper directly-trained spiking neural networks. *arXiv preprint arXiv:2210.06386*.
- Fu, X.; Zhang, J.; Meng, Z.; and King, I. 2020. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In *Proceedings of the web conference 2020*, 2331–2341.
- Gerstner, W.; Kistler, W. M.; Naud, R.; and Paninski, L. 2014. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press.
- He, X.; Deng, K.; Wang, X.; Li, Y.; Zhang, Y.; and Wang, M. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, 639–648.
- Hong, H.; Guo, H.; Lin, Y.; Yang, X.; Li, Z.; and Ye, J. 2020. An attention-based graph neural network for heterogeneous structural learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 4132–4139.
- Kim, S.; Park, S.; Na, B.; and Yoon, S. 2020. Spiking-yolo: spiking neural network for energy-efficient object detection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 11270–11277.
- Kim, Y.; Chough, J.; and Panda, P. 2022. Beyond classification: Directly training spiking neural networks for semantic segmentation. *Neuromorphic Computing and Engineering*, 2(4): 044015.
- Kipf, T. 2016. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv preprint arXiv:1609.02907*.
- Li, C.; Fu, J.; Yan, Y.; Zhao, Z.; and Zeng, Q. 2024. Higher order heterogeneous graph neural network based on node attribute enhancement. *Expert Systems with Applications*, 238: 122404.
- Li, J.; Yu, Z.; Zhu, Z.; Chen, L.; Yu, Q.; Zheng, Z.; Tian, S.; Wu, R.; and Meng, C. 2023a. Scaling up dynamic graph representation learning via spiking neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, 8588–8596.
- Li, J.; Zhang, H.; Wu, R.; Zhu, Z.; Wang, B.; Meng, C.; Zheng, Z.; and Chen, L. 2023b. A graph is worth 1-bit spikes: When graph contrastive learning meets spiking neural networks. *arXiv preprint arXiv:2305.19306*.
- Liu, H.; Liu, J.; Tang, F.; Li, P.; Chen, L.; Yu, J.; Zhu, Y.; Gao, M.; Yang, Y.; and Hou, X. 2024. Graph contrastive learning for truth inference. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, 263–275. IEEE.
- Liu, M.; Gao, H.; and Ji, S. 2020. Towards deeper graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 338–348.
- Lu, Z.; Fang, Y.; Yang, C.; and Shi, C. 2024. Heterogeneous graph transformer with poly-tokenization. In *International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence.
- Ma, Y.; Yan, N.; Li, J.; Mortazavi, M.; and Chawla, N. V. 2024. Hetgpt: Harnessing the power of prompt tuning in pre-trained heterogeneous graph neural networks. In *Proceedings of the ACM Web Conference 2024*, 1015–1023.
- Mao, Q.; Liu, Z.; Liu, C.; and Sun, J. 2023. Hinormer: Representation learning on heterogeneous information networks with graph transformer. In *Proceedings of the ACM web conference 2023*, 599–610.
- Peng, L.; Yang, C.; Chen, Y.; and Liu, W. 2023. Predicting CircRNA-disease associations via feature convolution learning with heterogeneous graph attention network. *IEEE Journal of Biomedical and Health Informatics*, 27(6): 3072–3082.
- Peng, Q.; Cao, B.; Xie, X.; Ye, H.; Liu, J.; and Li, Z. 2025. LLMsRec: Large language model with service network augmentation for web service recommendation. *Knowledge-Based Systems*, 323: 113710.
- Rueckauer, B.; Lungu, I.-A.; Hu, Y.; and Pfeiffer, M. 2016. Theory and tools for the conversion of analog to spiking convolutional neural networks. *arXiv preprint arXiv:1612.04052*.
- Salinas, E.; and Sejnowski, T. J. 2002. Integrate-and-fire neurons driven by correlated stochastic input. *Neural computation*, 14(9): 2111–2155.
- Shi, C.; Li, Y.; Zhang, J.; Sun, Y.; and Yu, P. S. 2016. A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering*, 29(1): 17–37.
- Tavanaei, A.; Ghodrati, M.; Kheradpisheh, S. R.; Masquelier, T.; and Maida, A. 2019. Deep learning in spiking neural networks. *Neural networks*, 111: 47–63.
- 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, X.; Ji, H.; Shi, C.; Wang, B.; Ye, Y.; Cui, P.; and Yu, P. S. 2019. Heterogeneous graph attention network. In *The world wide web conference, 2022–2032*.

Wu, D.; Sun, W.; He, Y.; Chen, Z.; and Luo, X. 2024. Mkgfenn: A multimodal knowledge graph fused end-to-end neural network for accurate drug–drug interaction prediction. *38(9)*: 10216–10224.

Xu, S.; Yang, C.; Shi, C.; Fang, Y.; Guo, Y.; Yang, T.; Zhang, L.; and Hu, M. 2021. Topic-aware heterogeneous graph neural network for link prediction. In *Proceedings of the 30th ACM international conference on information & knowledge management*, 2261–2270.

Yang, Y.; Guan, Z.; Li, J.; Zhao, W.; Cui, J.; and Wang, Q. 2021. Interpretable and efficient heterogeneous graph convolutional network. *IEEE Transactions on Knowledge and Data Engineering*, 35(2): 1637–1650.

Yin, N.; Wang, M.; Chen, Z.; De Masi, G.; Xiong, H.; and Gu, B. 2024. Dynamic spiking graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 16495–16503.

Yun, S.; Jeong, M.; Kim, R.; Kang, J.; and Kim, H. J. 2019. Graph transformer networks. *Advances in neural information processing systems*, 32.

Zhang, W.; Li, X.; and Nejdil, W. 2024. Adversarial mask explainer for graph neural networks. In *Proceedings of the ACM Web Conference 2024*, 861–869.

Zhao, M.; and Jia, A. L. 2024. Dahgn: Degree-aware heterogeneous graph neural network. *Knowledge-Based Systems*, 285: 111355.

Zhou, A.; Yang, J.; Qi, Y.; Shi, Y.; Qiao, T.; Zhao, W.; and Hu, C. 2023. Hardware-aware graph neural network automated design for edge computing platforms. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*, 1–6. IEEE.

Zhou, Z.; Zhu, Y.; He, C.; Wang, Y.; Yan, S.; Tian, Y.; and Yuan, L. 2022. Spikformer: When spiking neural network meets transformer. *arXiv preprint arXiv:2209.15425*.

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